View, add and edit your notes in the app

Angular Router and Route Guards | Angular Concepts made easy | Procademy Classes

Generated on February 21, 2024

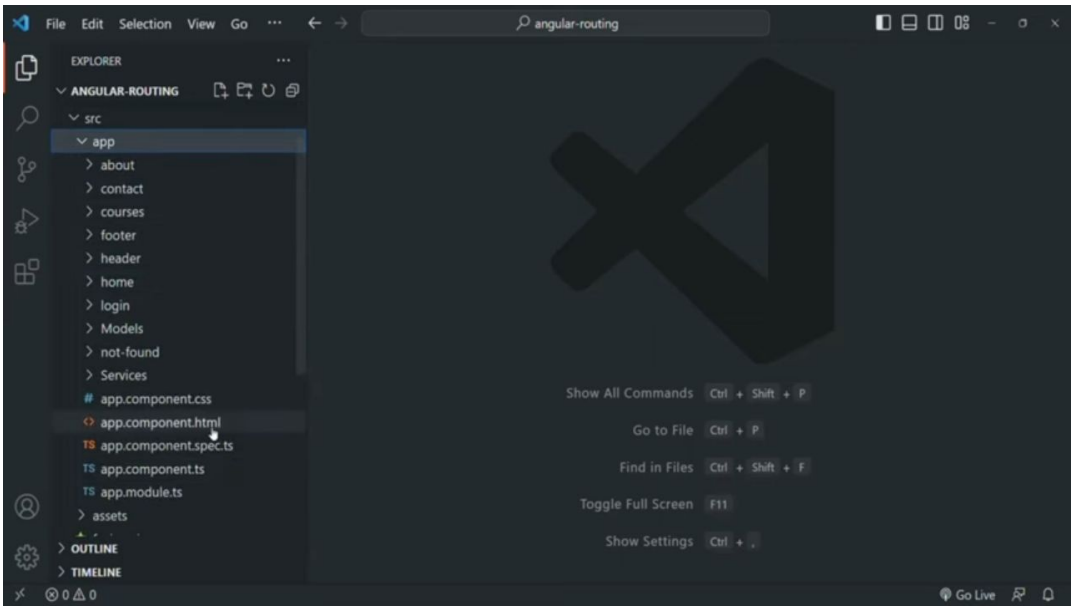
Summary

Notes	Screenshots	Bookmarks
0	190	0

Angular routing allows for seamless navigation between different components in an application using the angular/router module.

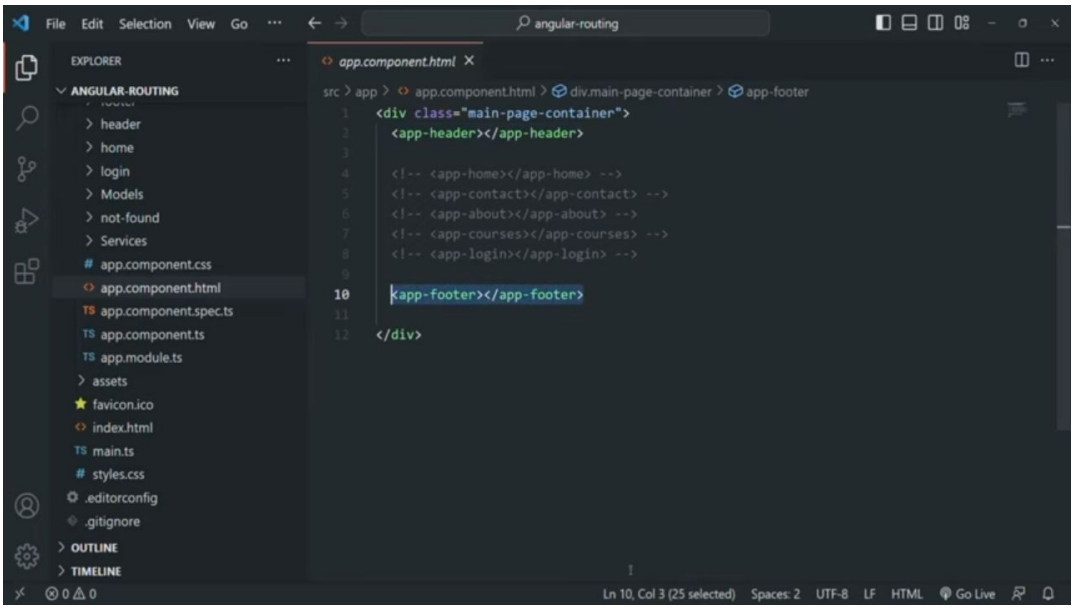
0:05

0:15



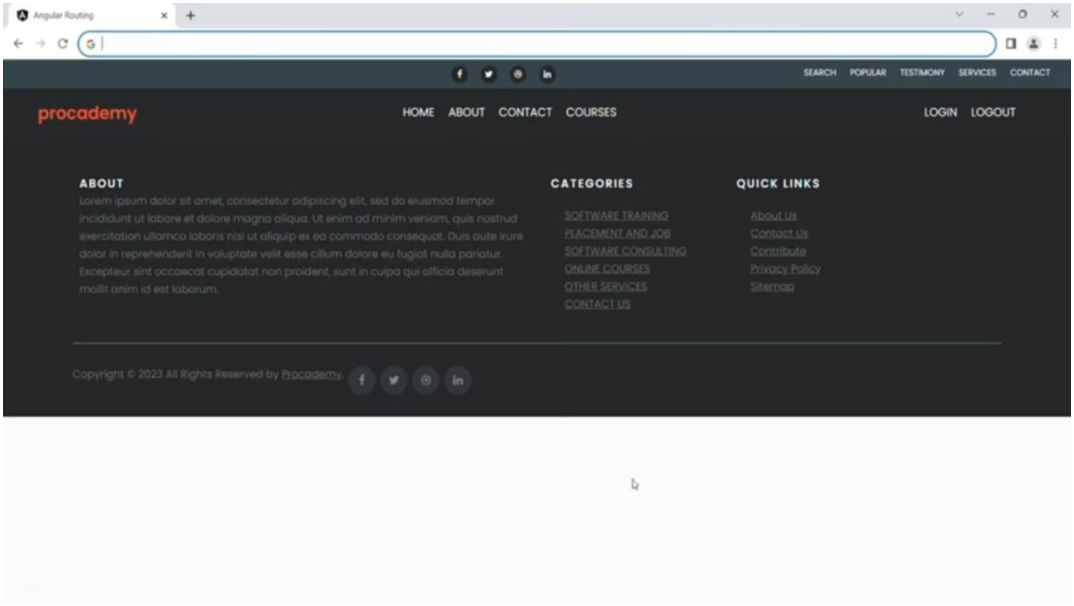
In the Source folder, the app component serves as the root component and utilizes selectors for both header and footer components.

1:10



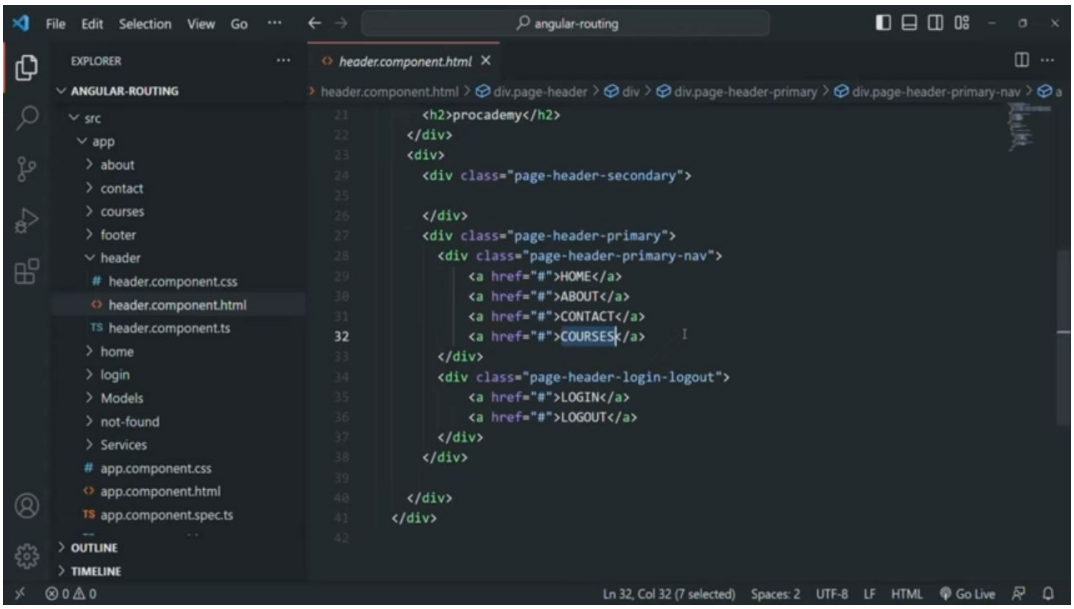
App component is the root component used in index.html file, rendering the content of header and footer components.

1:30



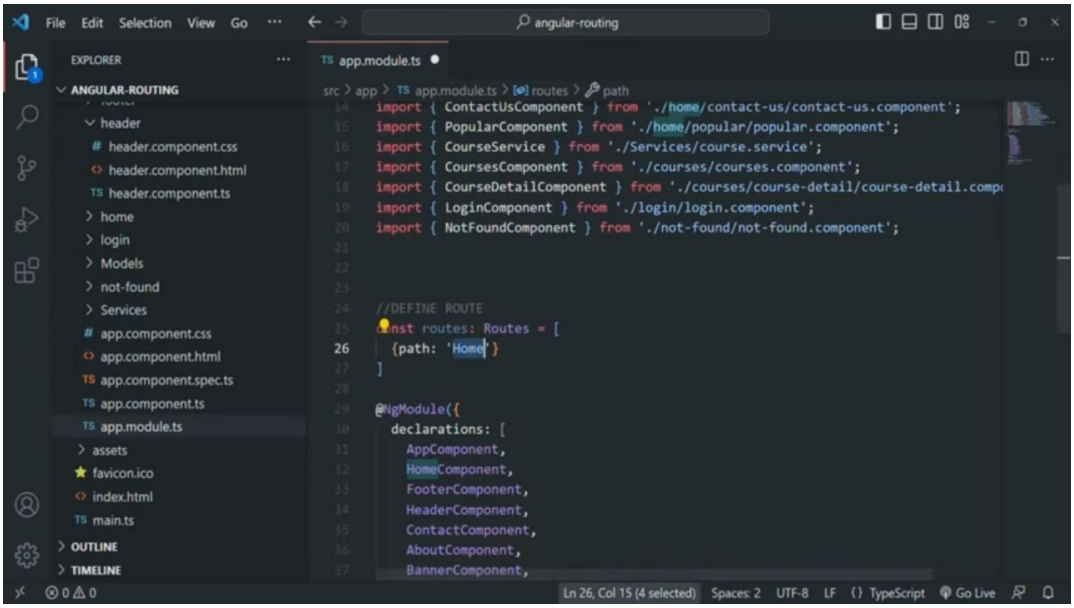
Our current focus is on the main pages of our angular application, including home, about, contact, and courses links.

2:28



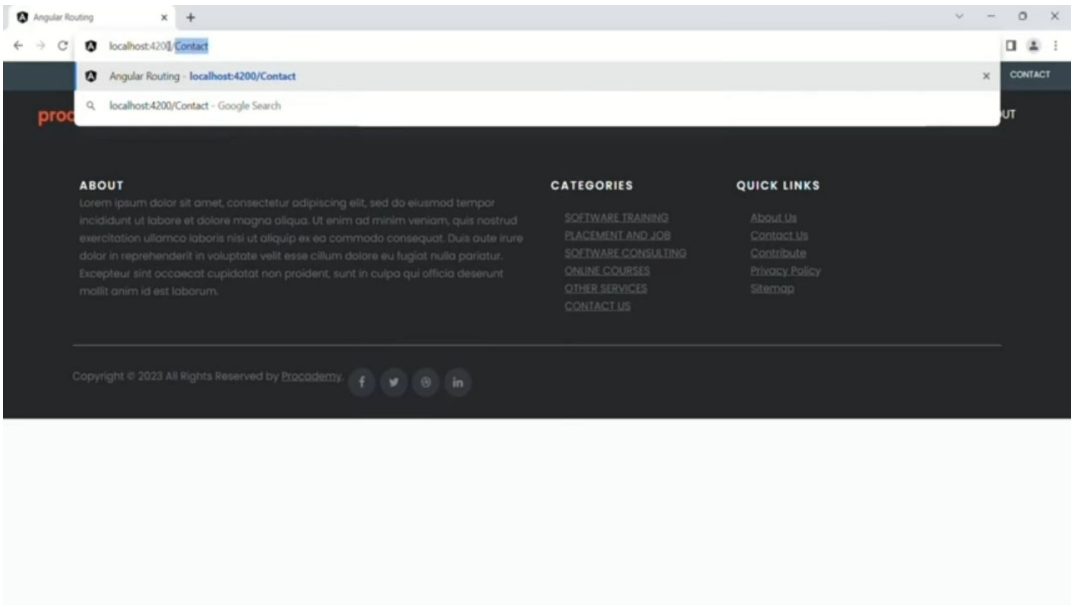
A new web page has been designed to display different components when specific links are clicked.

2:53



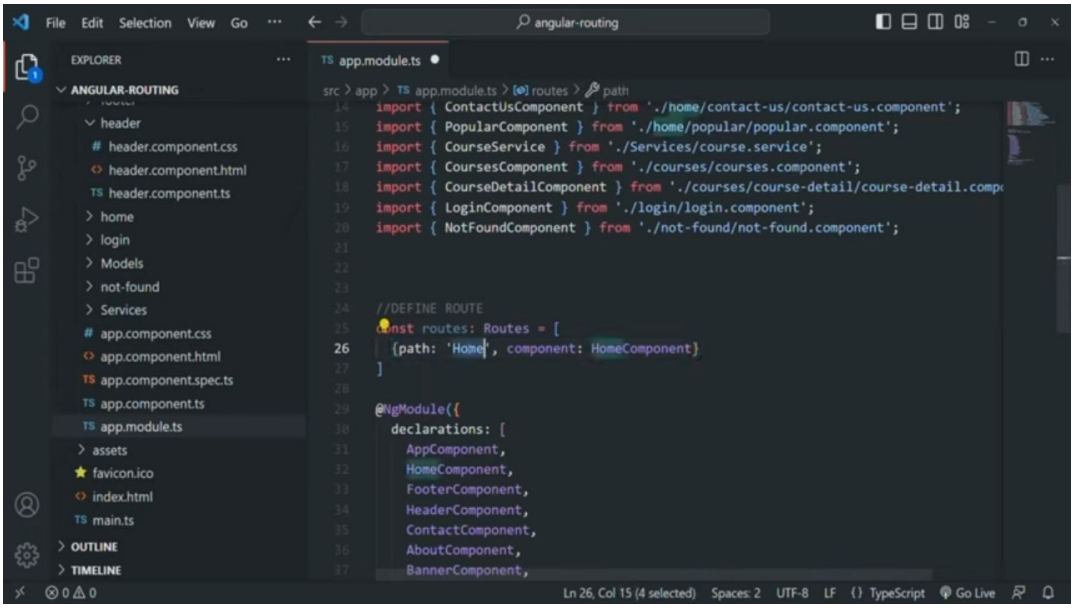
In the app module, routes are defined directly before the NG module directive.

3:56



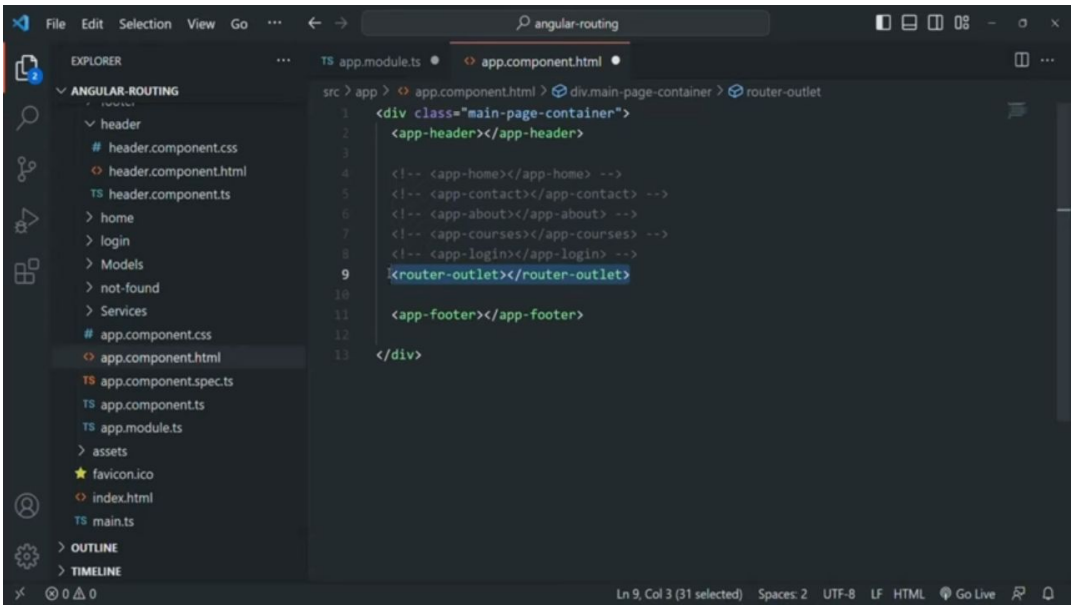
Understanding the difference between URL and path is crucial for navigating websites effectively.

5:55



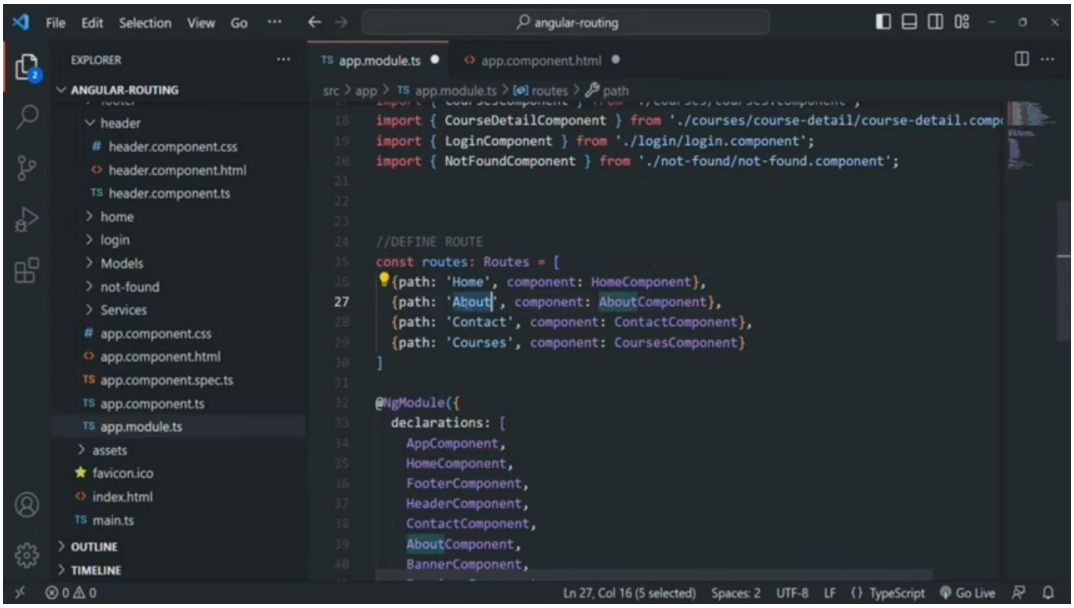
When a user types "home" in the address bar, the web page will display the view of the Home component.

6:20

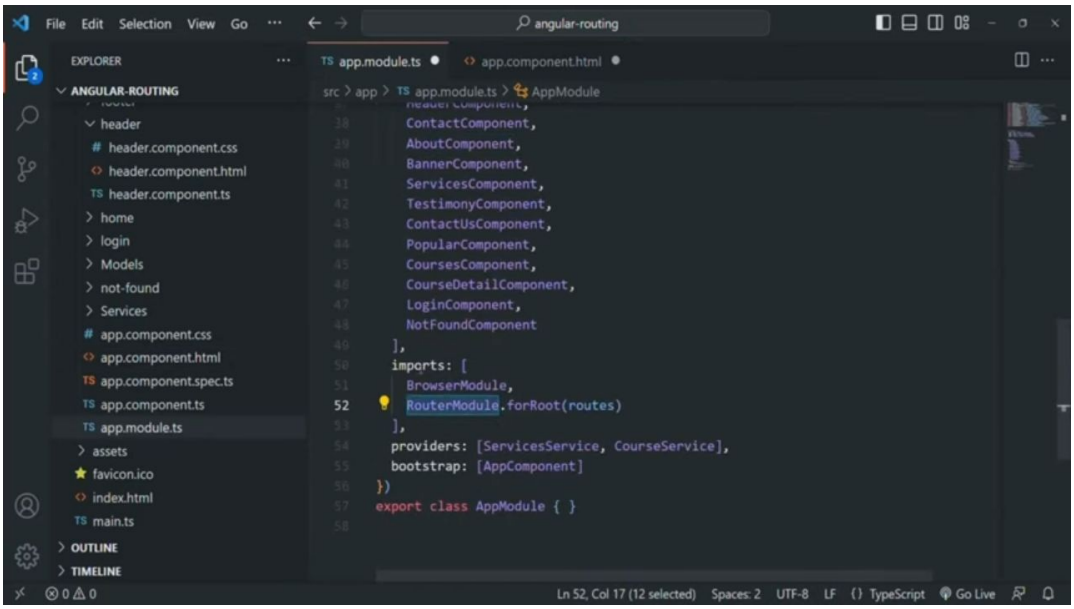


Angular uses a directive called router Outlet to render view templates between the header and footer components based on the URL path.

10:34

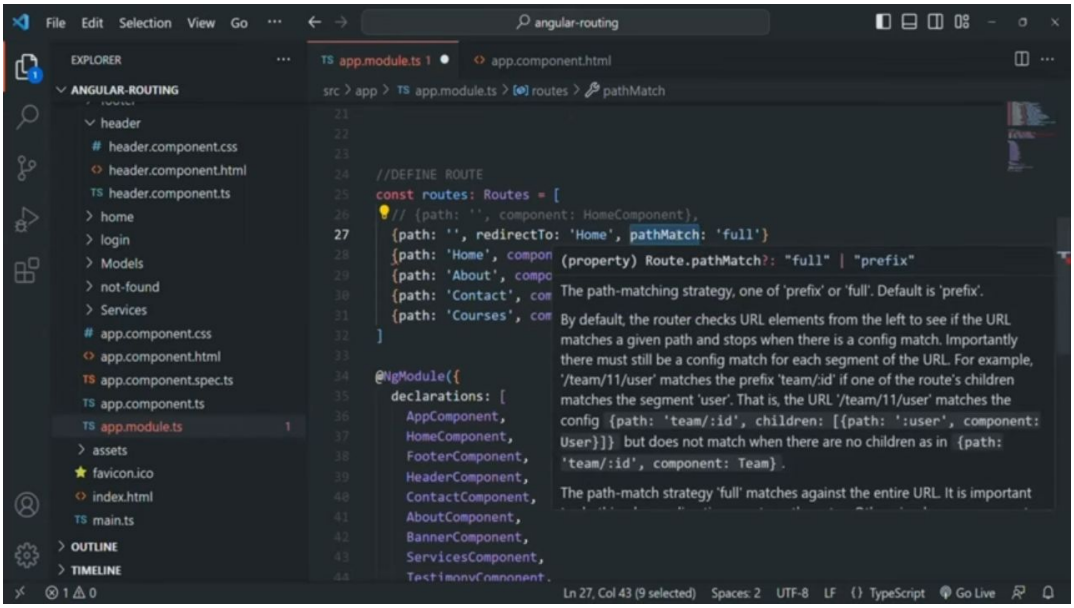


10:43



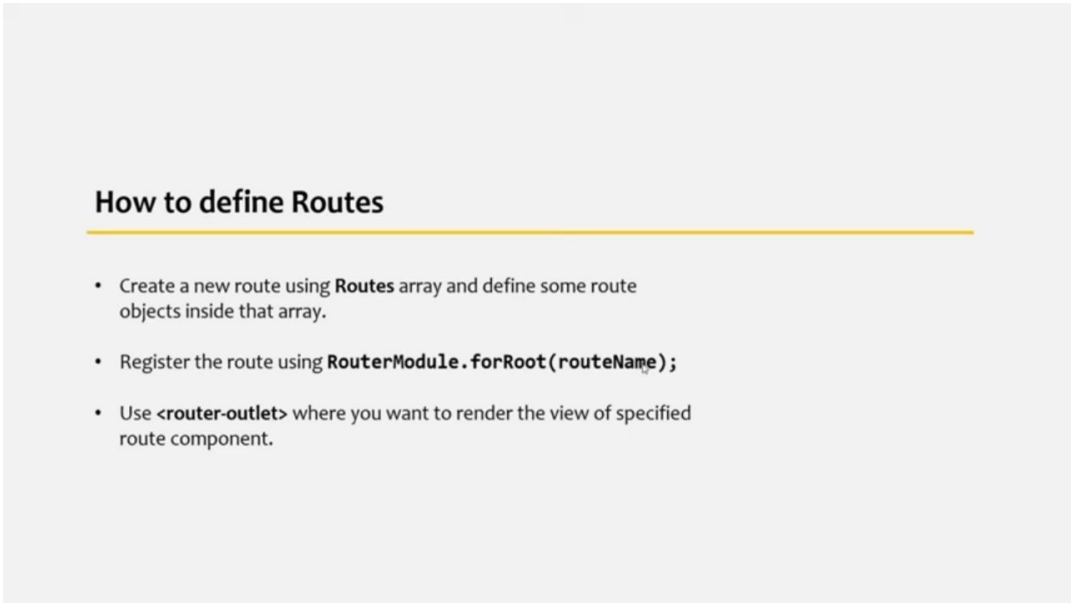
The router Outlet directive is automatically imported when the router module is imported, eliminating the need for explicit import statements.

11:16



Angular's path mesh property allows for easy configuration of URL matching and redirection.

15:26

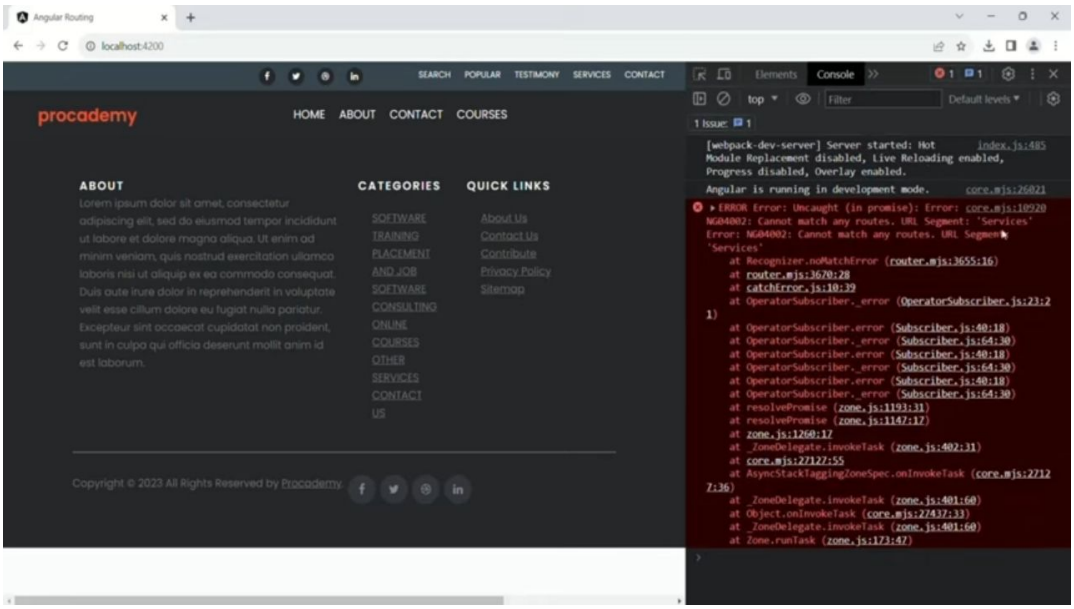


Angular application now has five implemented routes for routing.

16:50

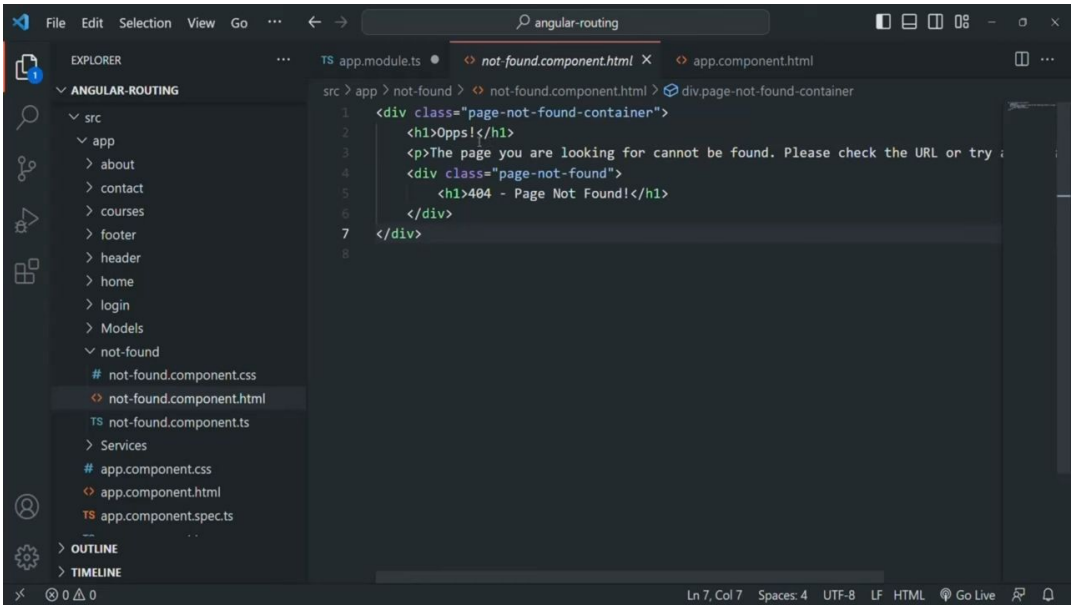
Angular application now includes five implemented routes for Home, About, Contact, and Courses.

16:58



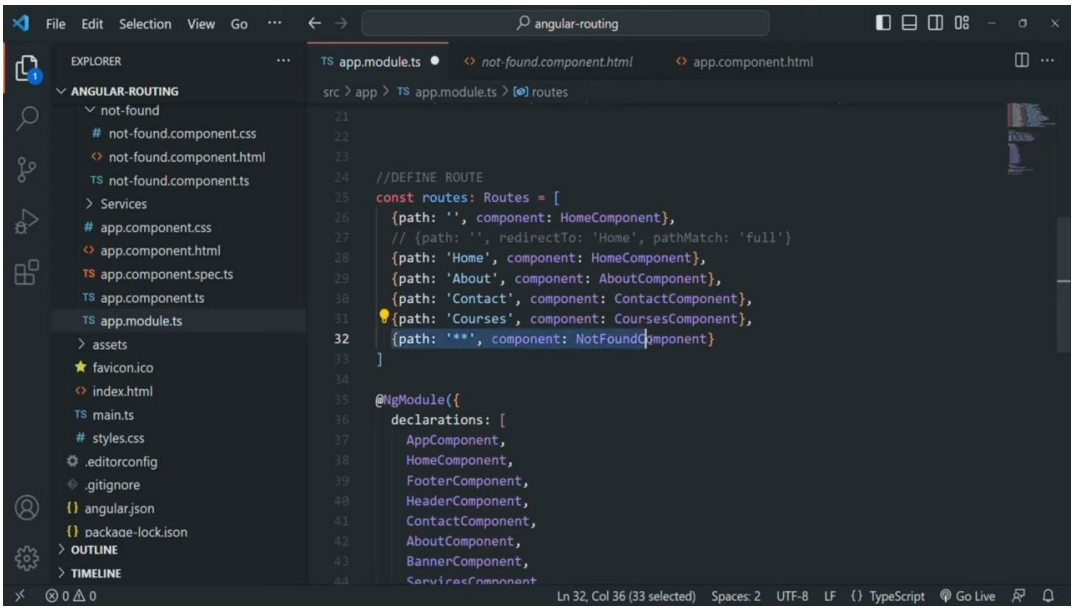
Developer console reveals error in website routing, causing redirection to root URL.

18:28



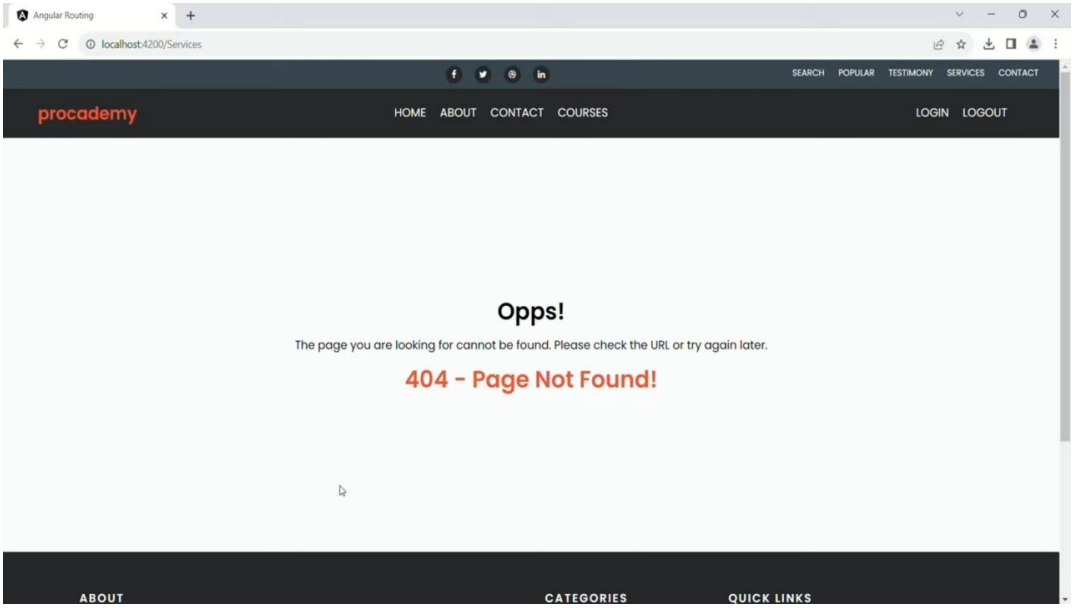
Developing a custom 404 page for your website can help improve user experience and provide helpful information when a page cannot be found.

20:26



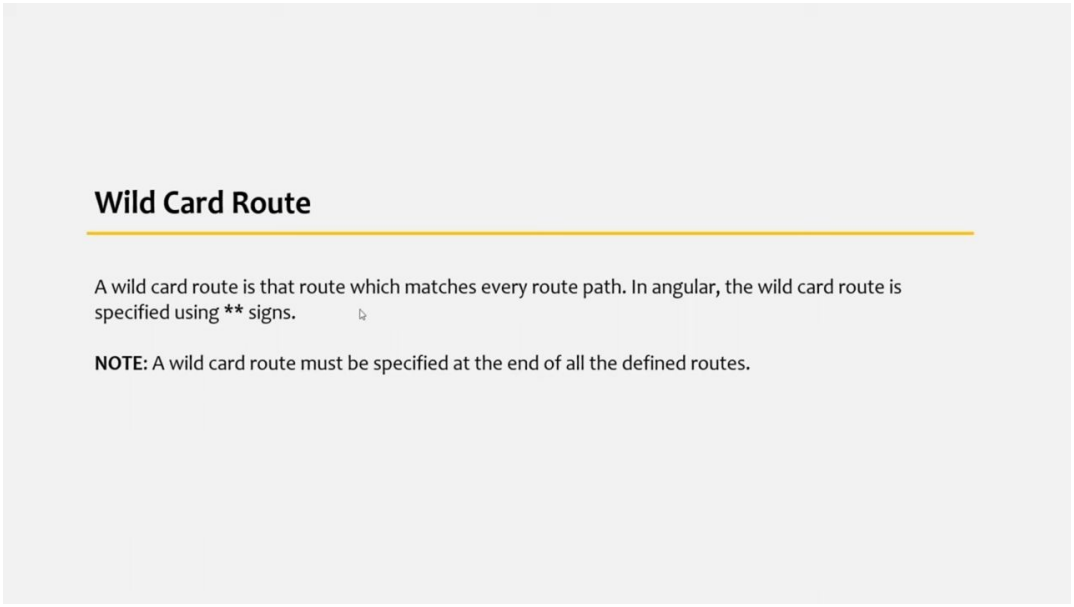
New routing component allows for dynamic rendering of not found views based on user input in the address bar.

21:11



The not found component view is rendered on the web page and no errors are present in the developer console.

22:40

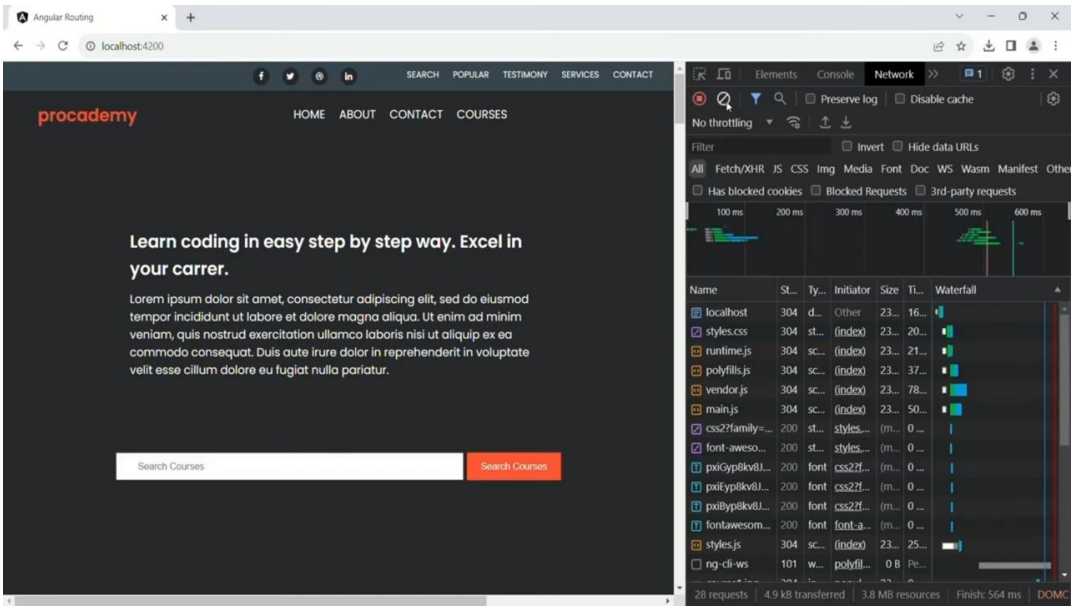


The wildcard route in Angular is used to match any route path and must be specified at the end of all defined routes.

24:01

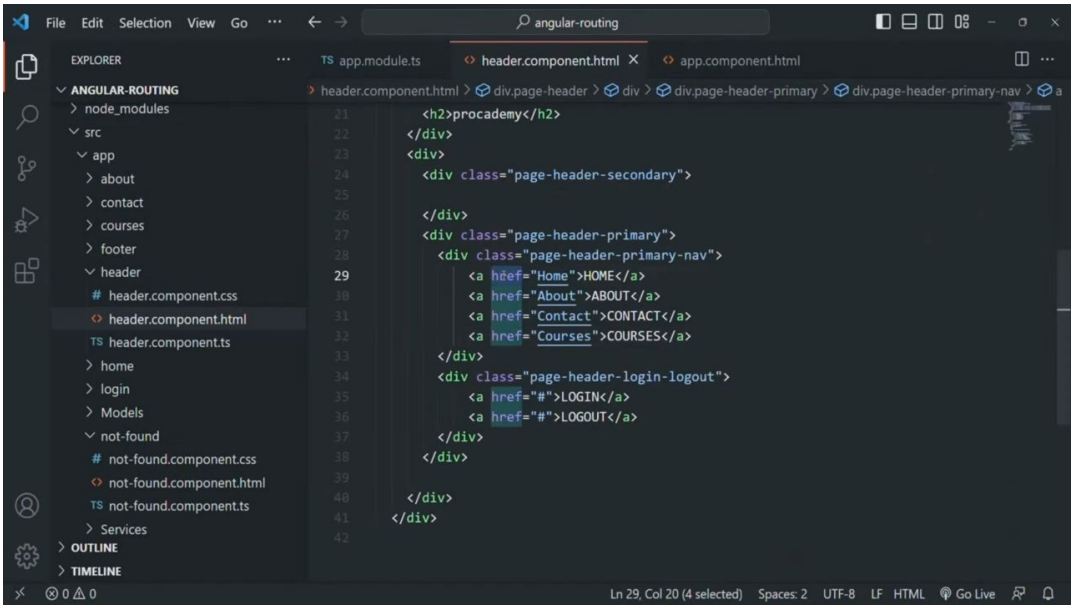
Developing functional navigation links in an Angular application is essential for displaying the correct views when users click on specific links.

28:57



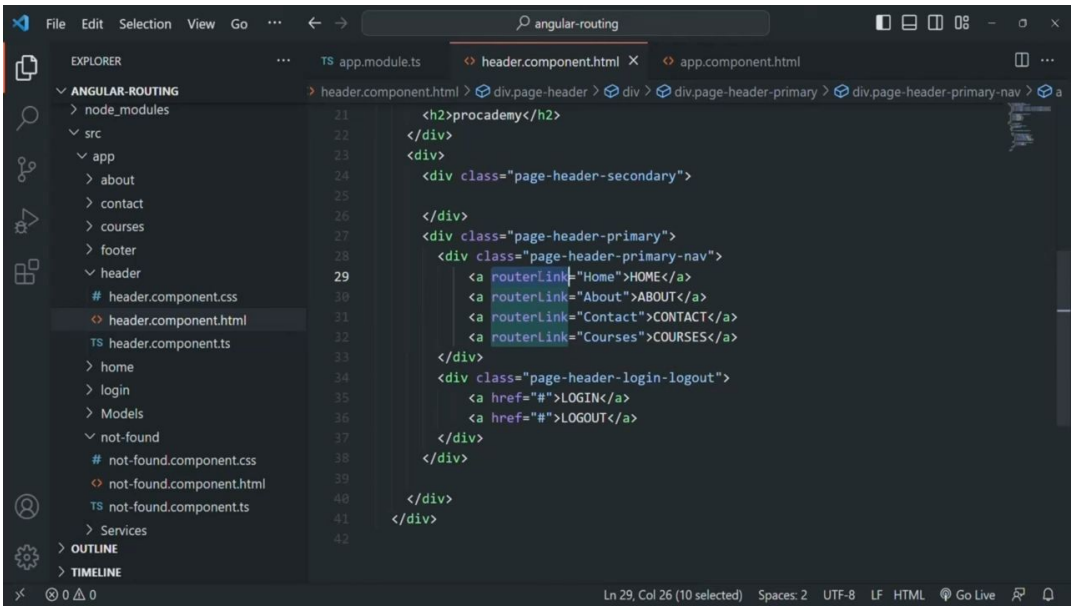
Runtime.js, polyfill.js, and vendor.js1del bundles have been downloaded for the application, but files are being re-downloaded when navigating to different pages.

33:16



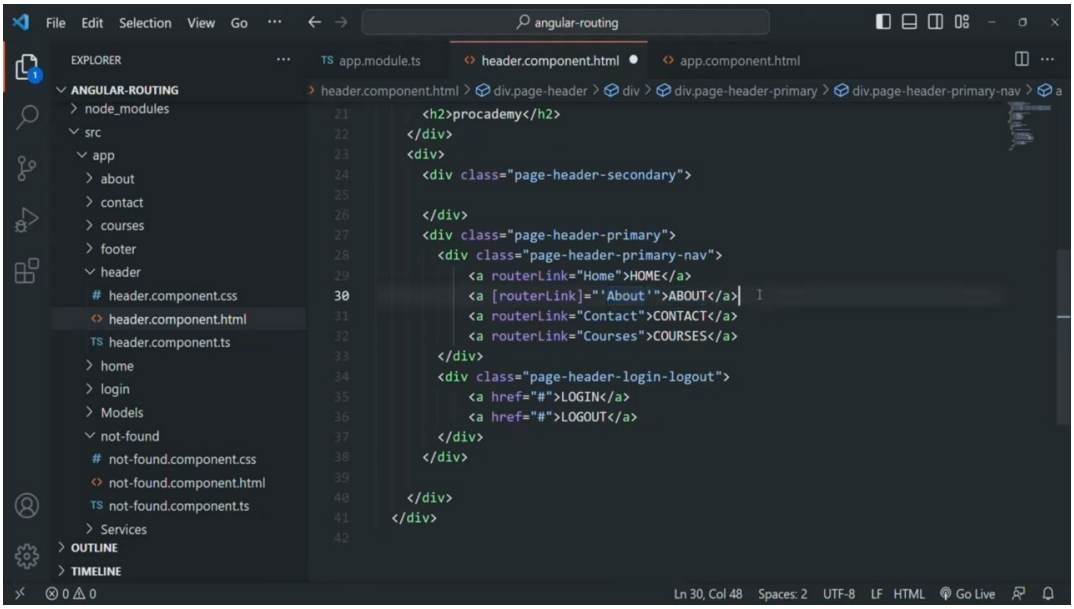
Developing Angular applications requires proper use of directives to ensure desired behavior.

34:23



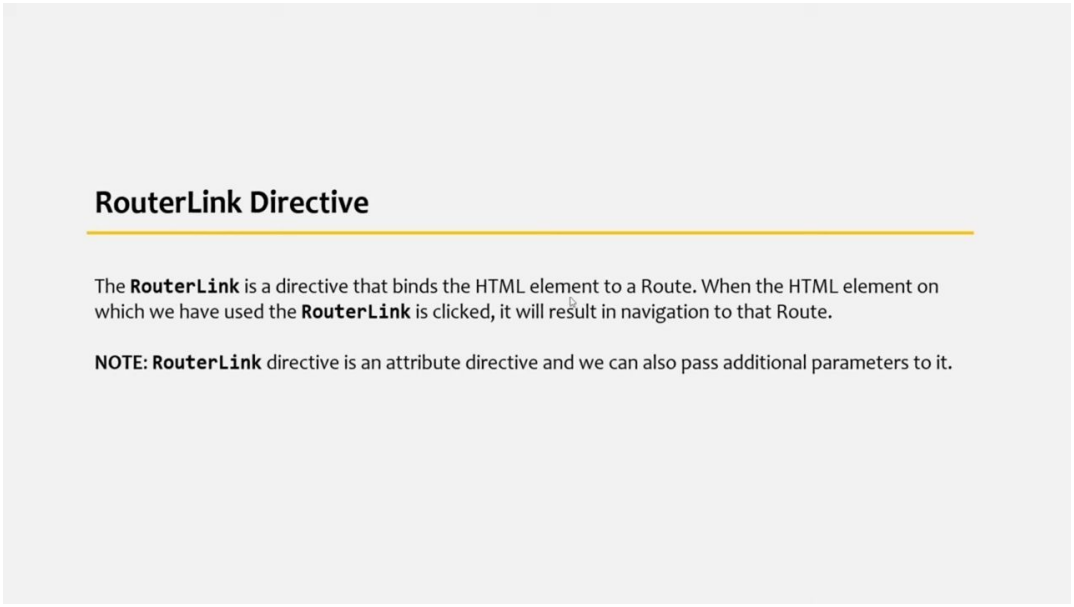
Using router links in applications can prevent the need to constantly download the complete application file when navigating.

36:43



Square brackets cannot be used to assign a string value, instead the value must be wrapped in single quotes.

37:35

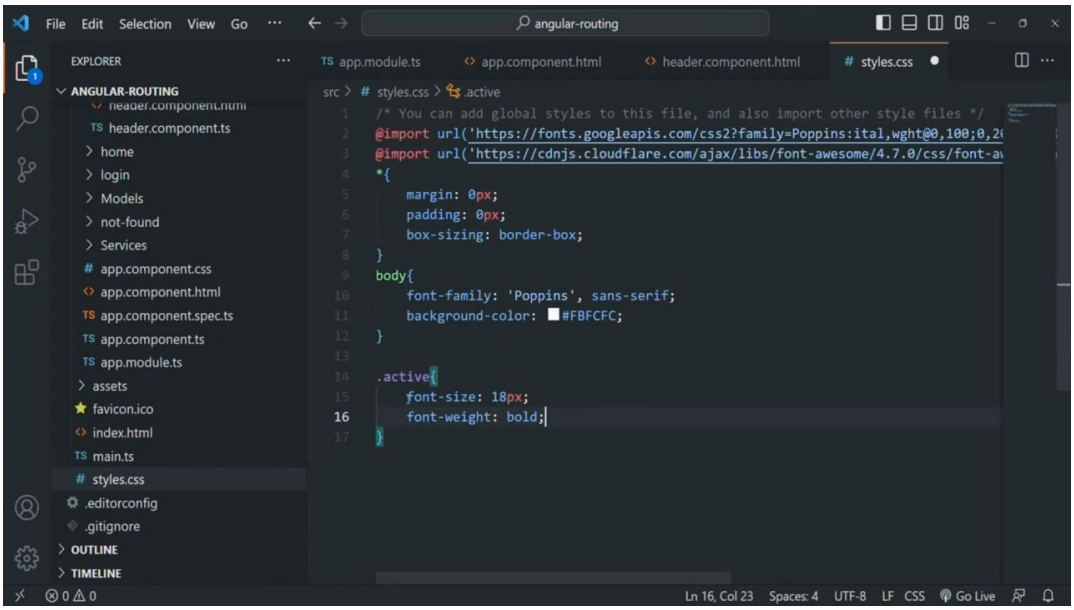


The router link directive in our example binds HTML elements to specific routes, allowing for easy navigation within the website.

37:56

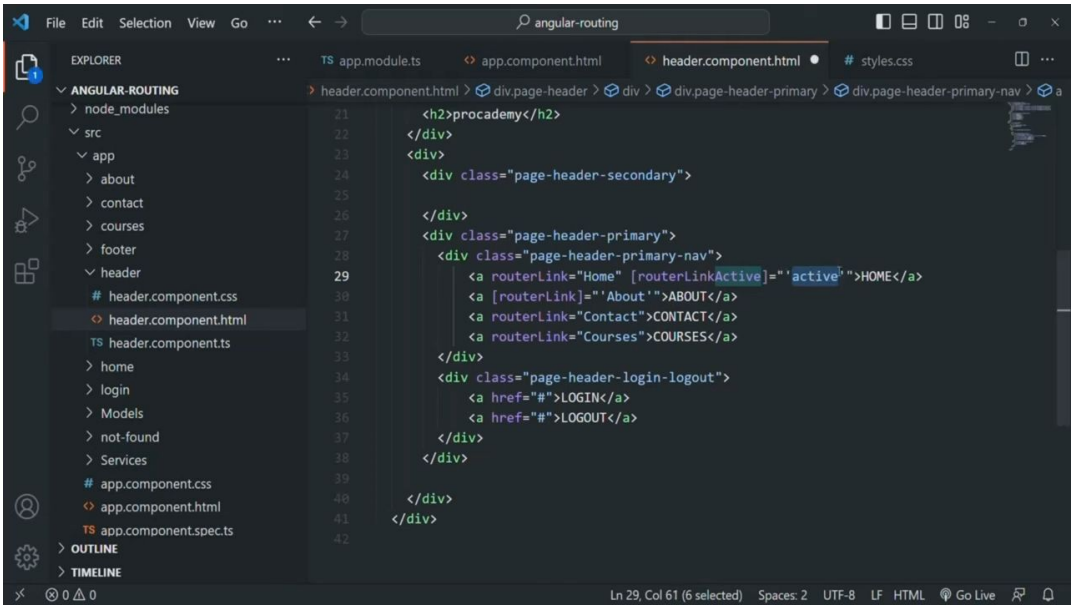
The implementation of routing in our Angular application has allowed for dynamic content changes and seamless navigation through the use of links.

39:40



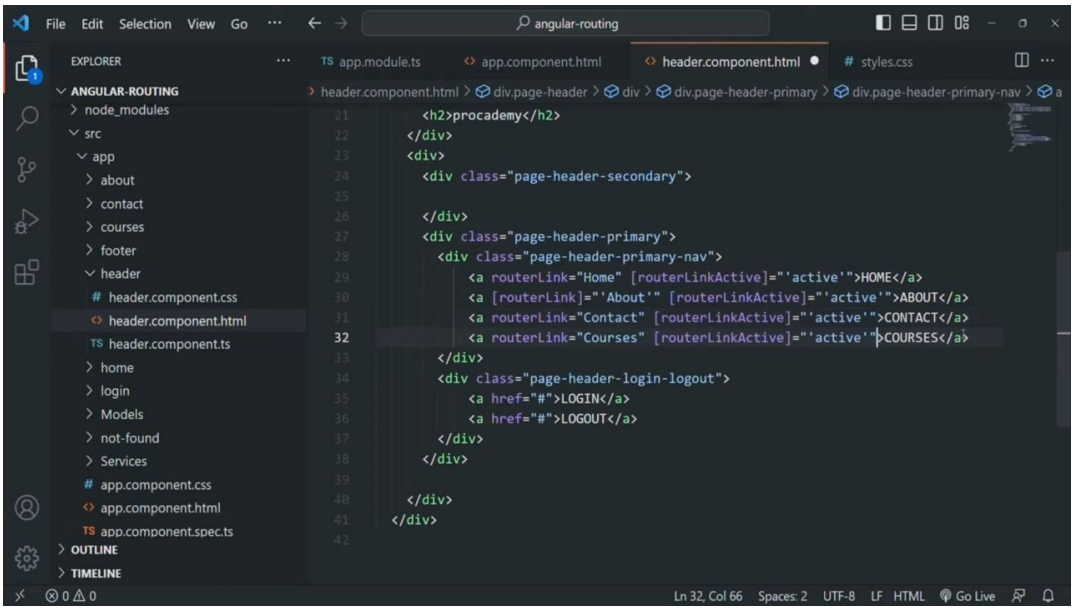
In the style.css file, a new CSS class called "active" has been created to apply font size and weight to active links.

41:16



The router link active directive requires single quotes to assign a string value to the attribute.

42:42



Website developer enhances user experience by highlighting active links with larger font size and bold text.

43:04

RouterLinkActive Directive

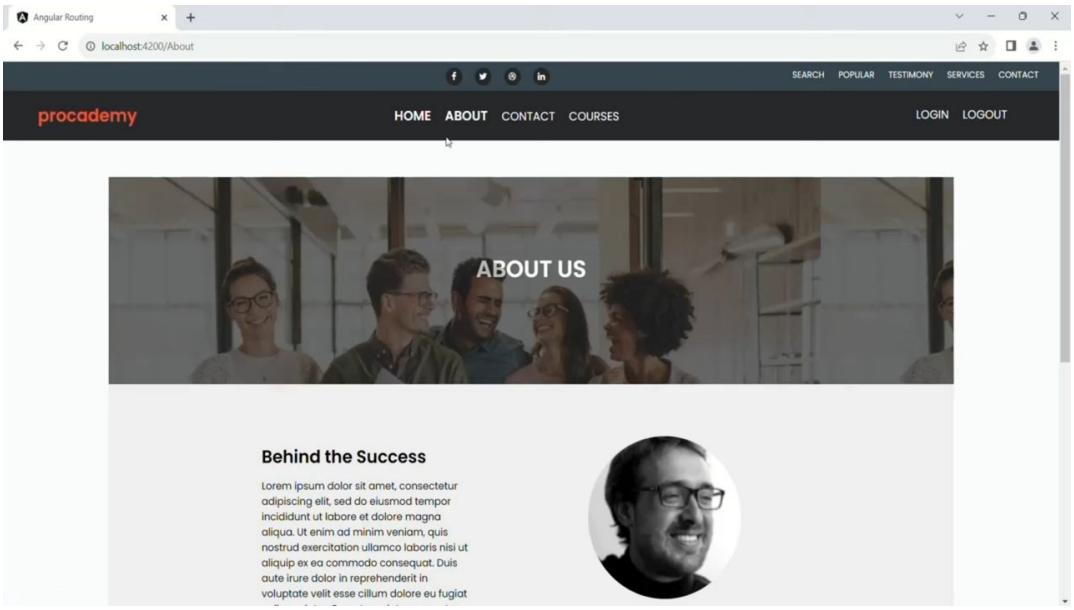
The **RouterLinkActive** is a directive for adding or removing classes from an HTML element that is bound to **RouteLink**.

Using **RouterLinkActive** directive, we can toggle CSS classes for active route link based on the current router state.

The main use case of **RouterLinkActive** directive is to highlight which route is currently active.

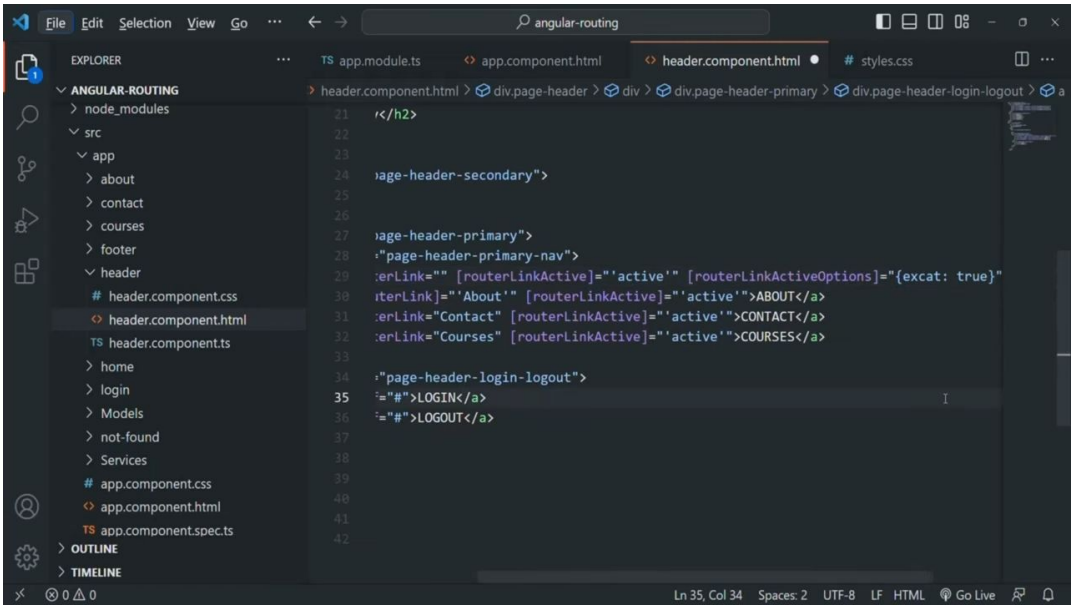
The router link active directive is a useful tool for highlighting the active route in an Angular application.

43:37



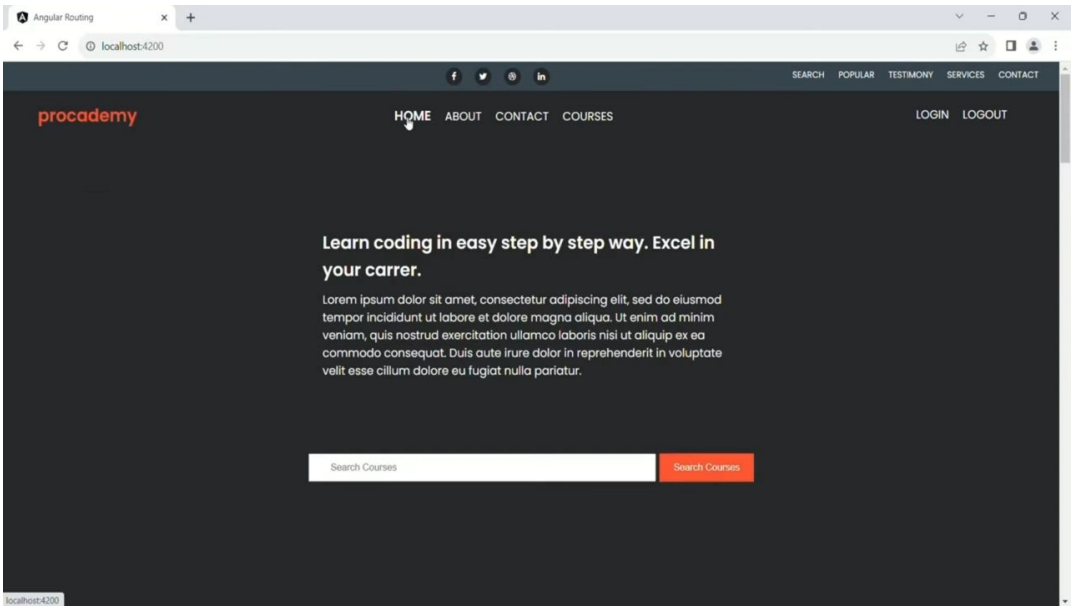
The about page has been rendered and is now the active link, but the home page is also shown as active due to its relationship as a child link.

45:26



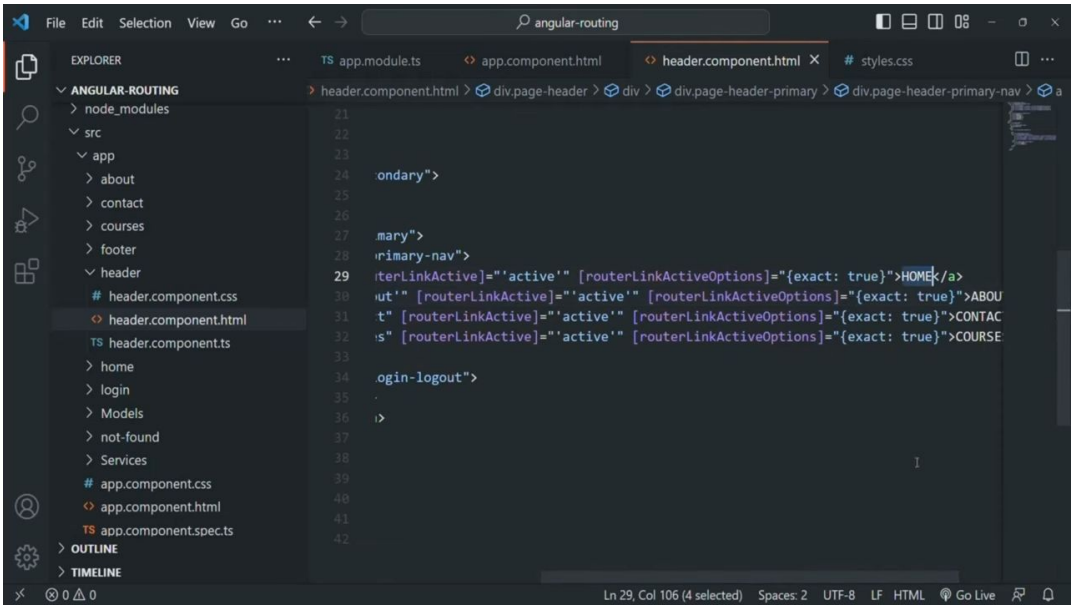
Using the attribute directive, we can assign an object to ensure that the active CSS class only gets applied to routes whose path matches the current URL.

47:17



Web page styling issue resolved with new route changes.

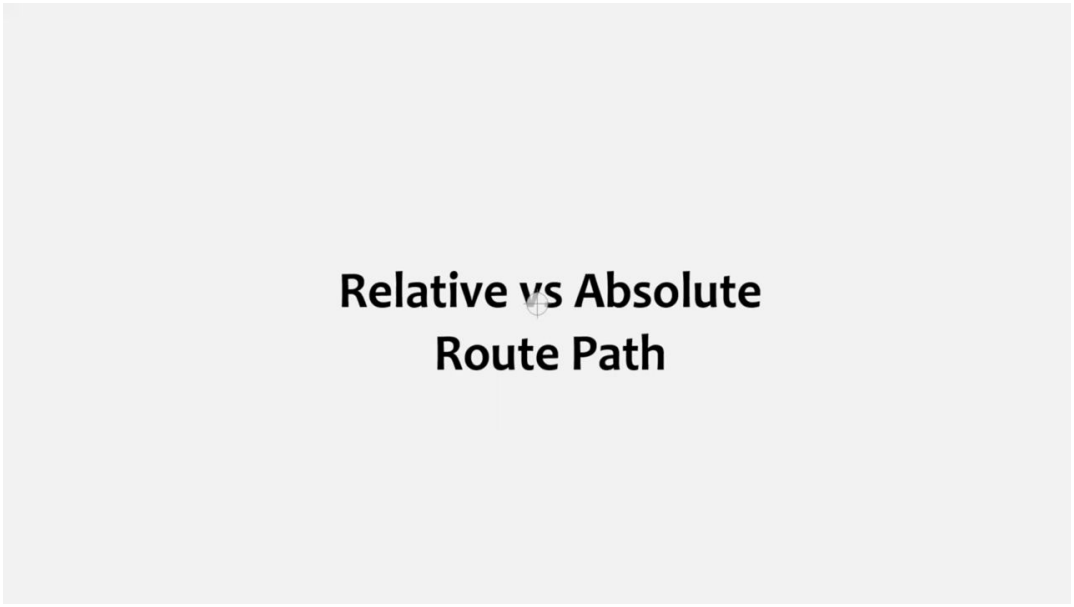
48:30



The issue of styling has been resolved in the lecture, with a focus on changing the path for the home link.

48:45

48:55



Understanding the difference between relative and absolute route paths is crucial for navigating an Angular application effectively.

49:29

Absolute Path

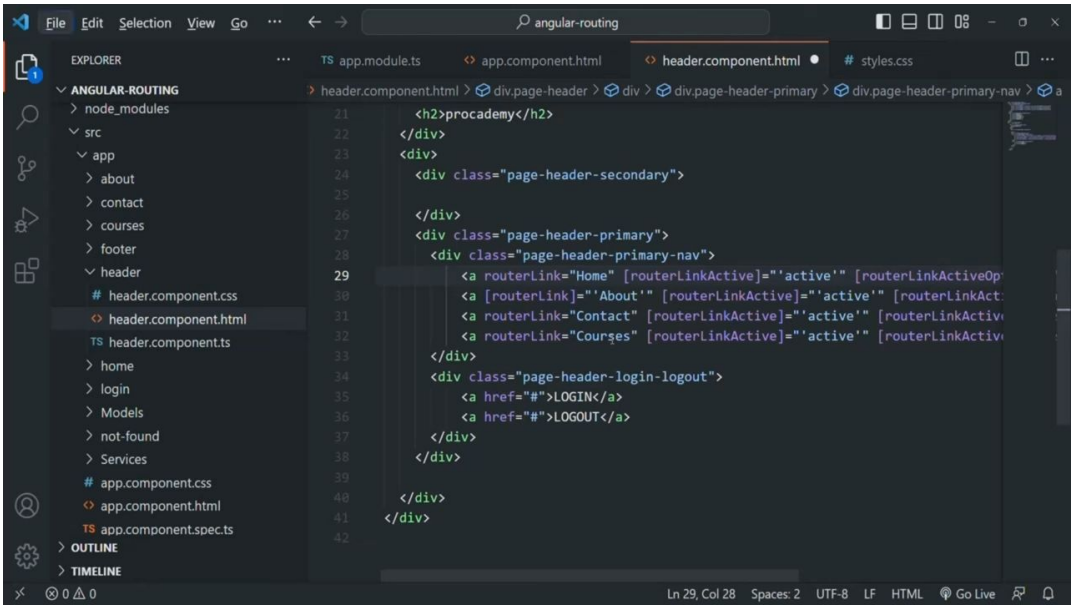
When we use a slash (“/”) before the router link path, in that case it uses absolute path and the path is directly appended to root url.

```
<a routerLink=“/About”>About</a>

URL: localhost:4200/About
```

When creating links in web development, the path is directly appended to the root URL.

50:53



Understanding the use of slashes in file paths is crucial for navigating and linking within a website.

51:37

Relative Path

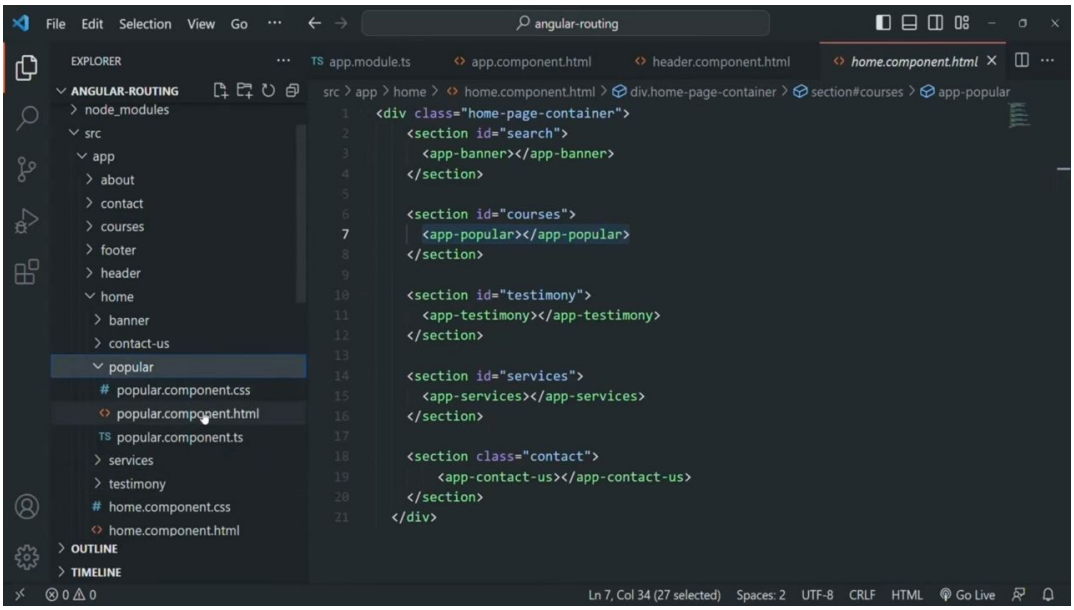
When we do not use a slash (“/”) before the router link path, in that case it uses relative path and the path is appended to the currently active route.

For example, let’s say the currently active route is **About**. And a link is defined in About page to go to home page as shown below. If that link is clicked, the path will be appended to currently active route.

```
<a routerLink="Home">Go to Home</a>
```

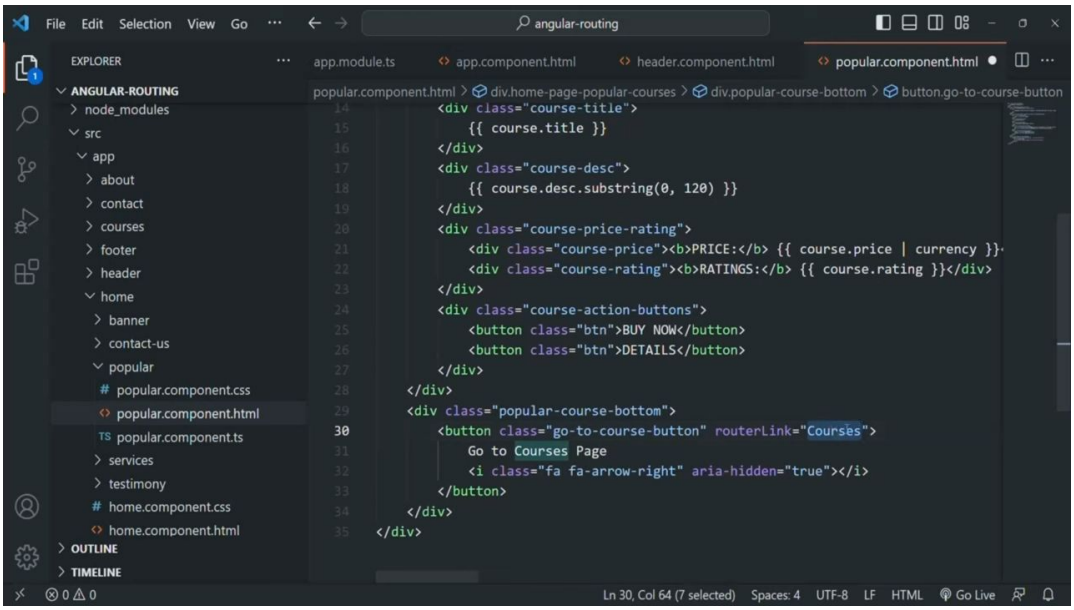
URL: `localhost:4200/About/Home`

51:51



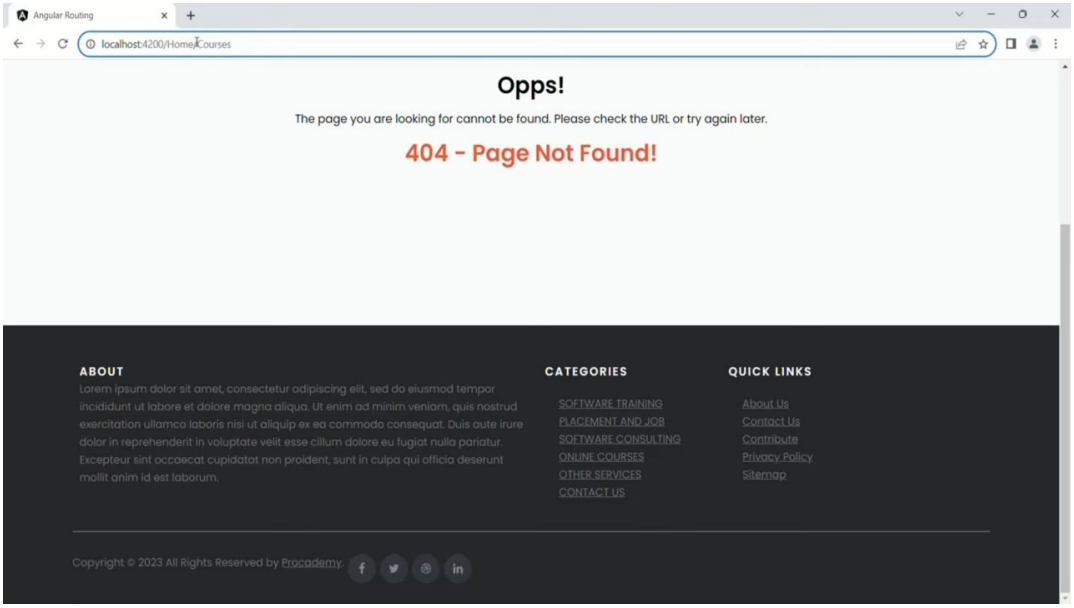
New app component "Popular" is being used to display popular causes and includes a button with a router link directive.

53:27



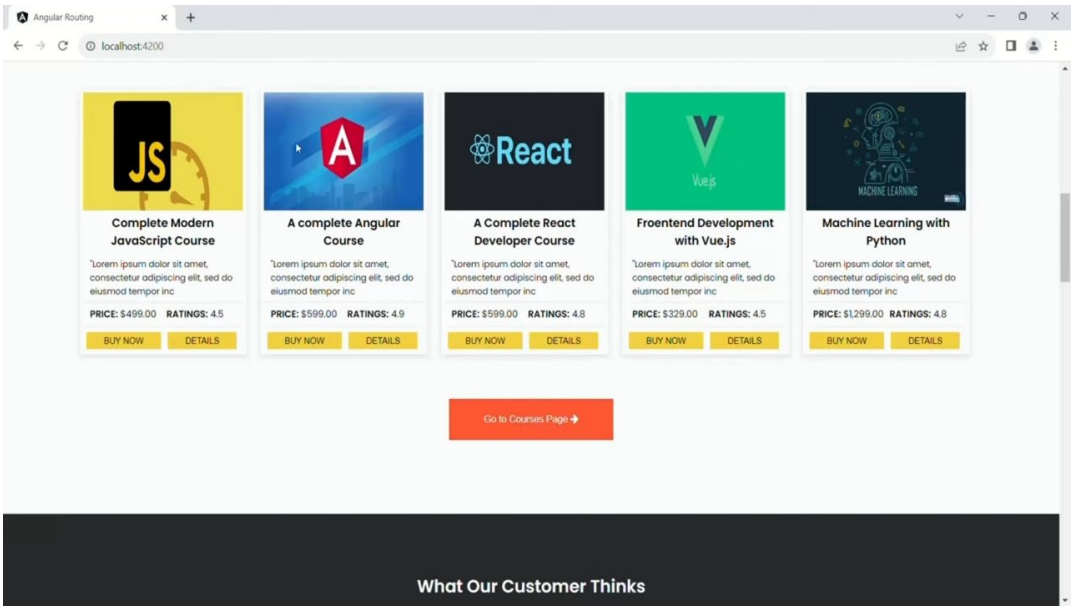
Learn how to use router links in HTML elements to redirect users to specific pages on your website.

54:04



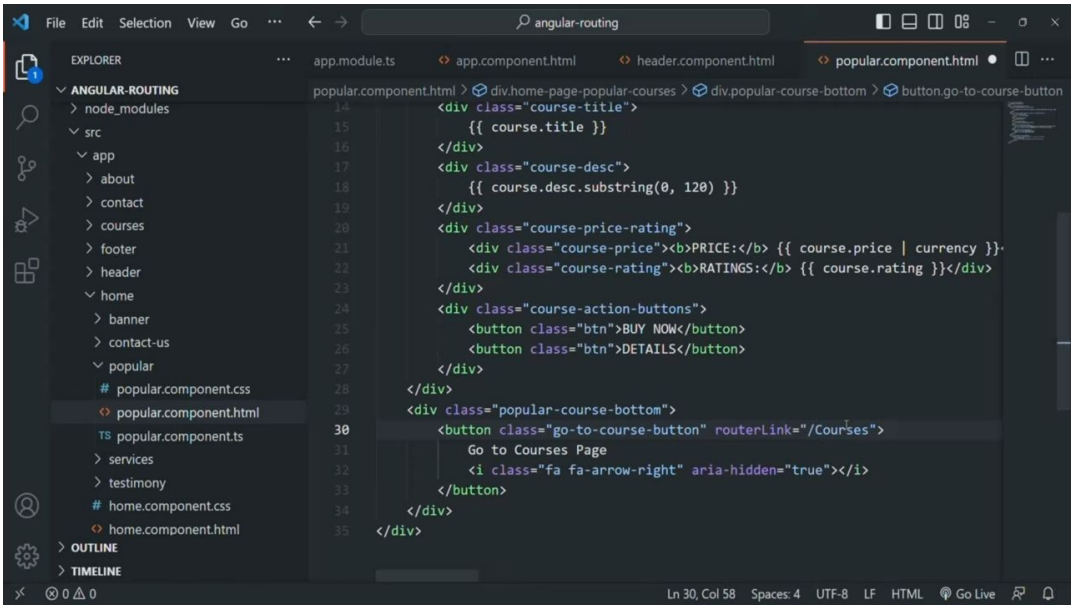
Using relative paths in URLs can lead to unexpected results when navigating through a website.

54:51



Navigating through the website's routes and components is made easy with the seamless appending of paths.

55:25



Using absolute paths in web development ensures that the specified URL will always be appended to the root URL, regardless of the current active route.

56:17

Relative Path

When we use a dot & a slash (“./”) before the router link path, in that case it uses relative path and the path is appended to the currently active route.

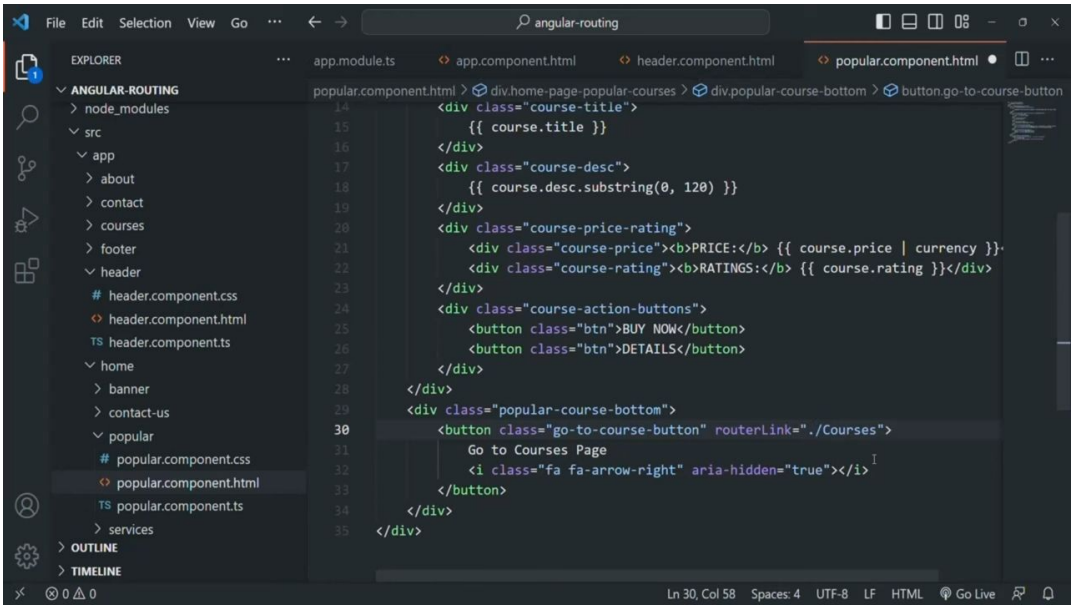
Current Active Router Link Path: **localhost:4200/Books/Author**

```
<a routerLink="../Stephen-King">Stephen King</a>
```

URL: localhost:4200/Books/Author/Stephen-King

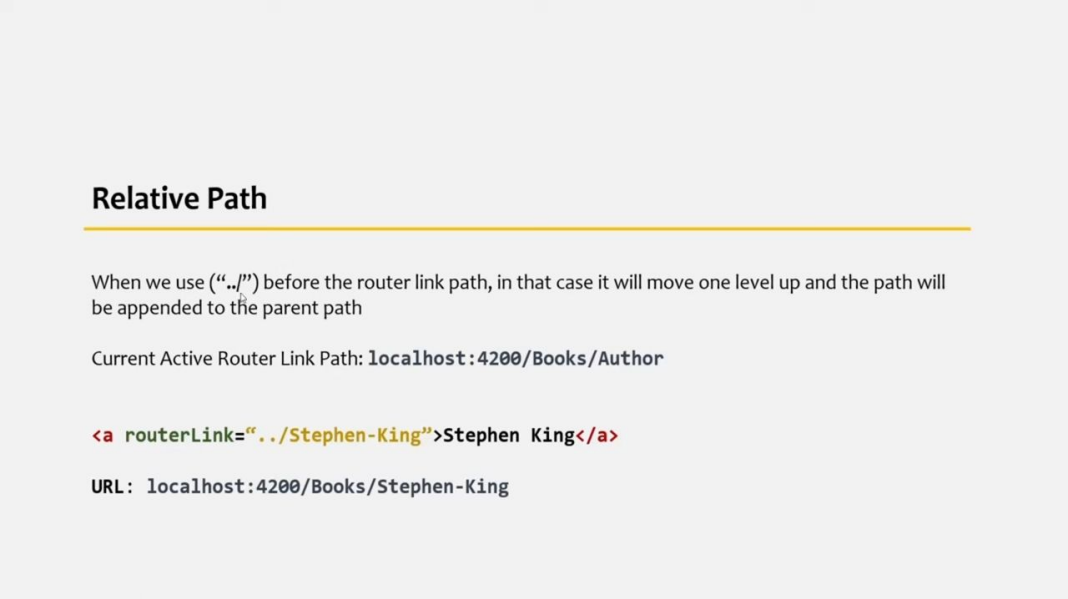
Understanding the difference between absolute and relative paths is crucial for navigating URLs effectively.

1:00:02



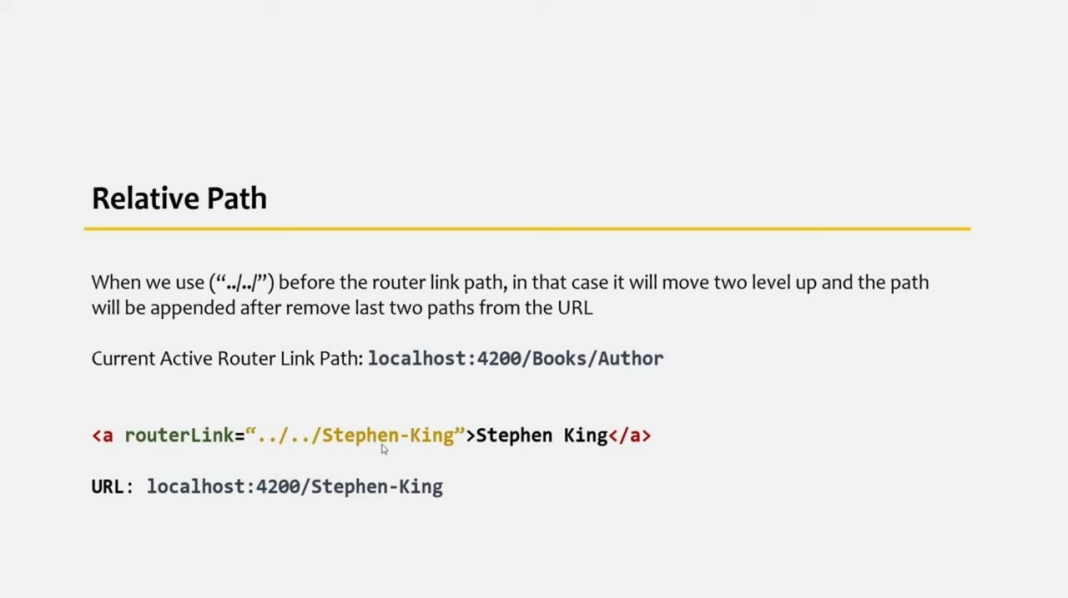
Understanding the difference between absolute and relative paths is crucial for navigating web pages effectively.

1:00:22



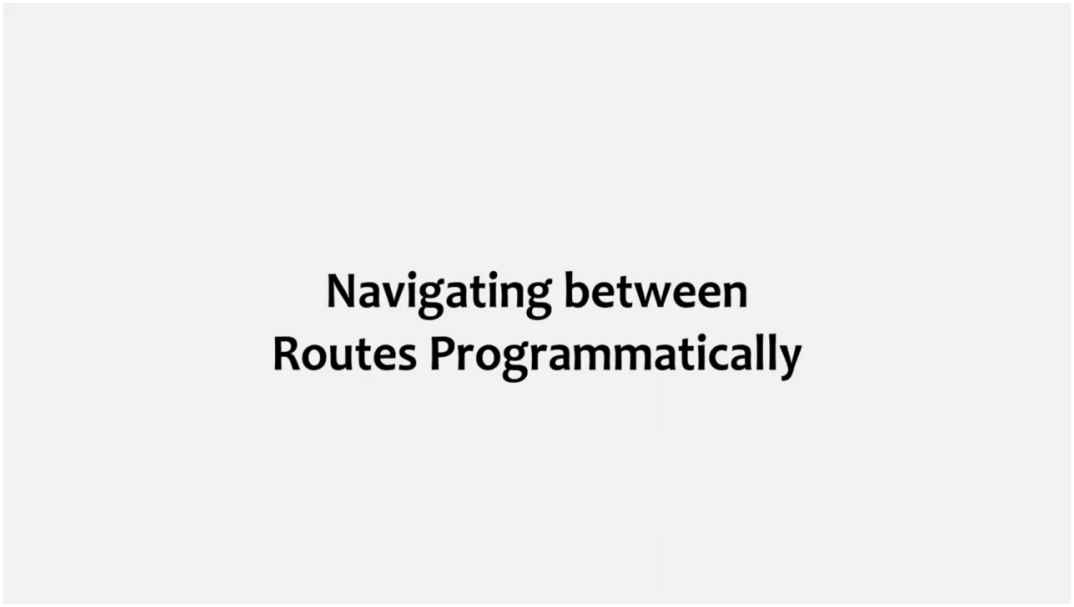
Understanding how to use two dot and a slash before the router link path can help navigate through URL paths more effectively.

1:01:16



URL paths can be manipulated to navigate to different levels and append new paths.

1:02:17

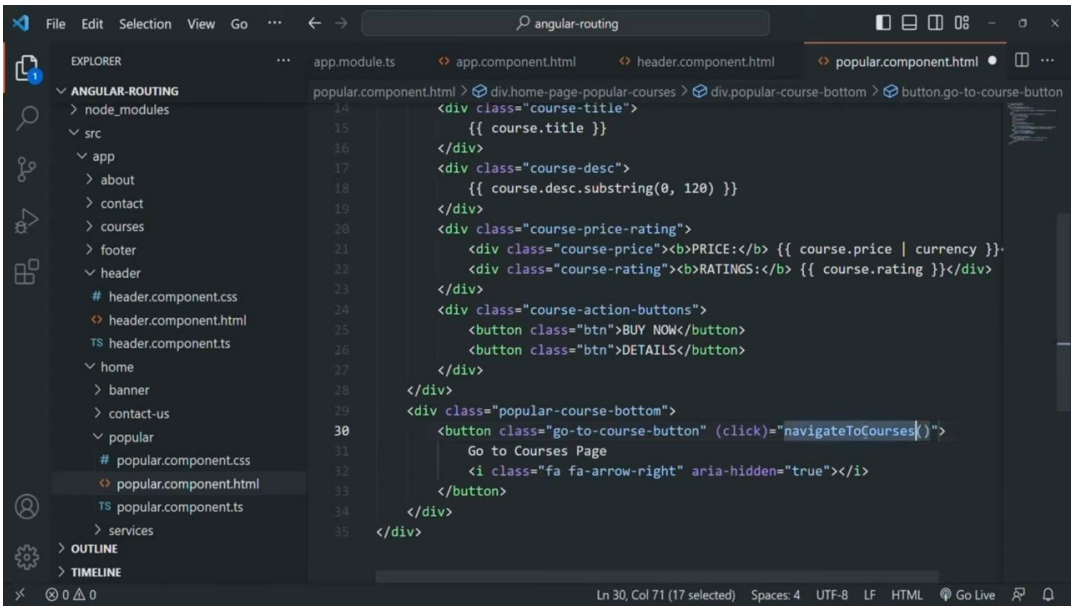


Absolute and relative paths are essential for defining router link paths in Angular, allowing for seamless navigation between routes.

1:02:56

In this lecture, you will learn how to navigate between routes programmatically in Angular.

1:03:13



A new method called "getCourses" has been assigned to the popular component class for retrieving courses.

1:04:35

navigate() Method

Using **navigate()** method, we can navigate from one route to another programmatically. The navigate method takes an array as an argument and in that array we can specify route segments.

Let's say, we want to navigate the user to following route:

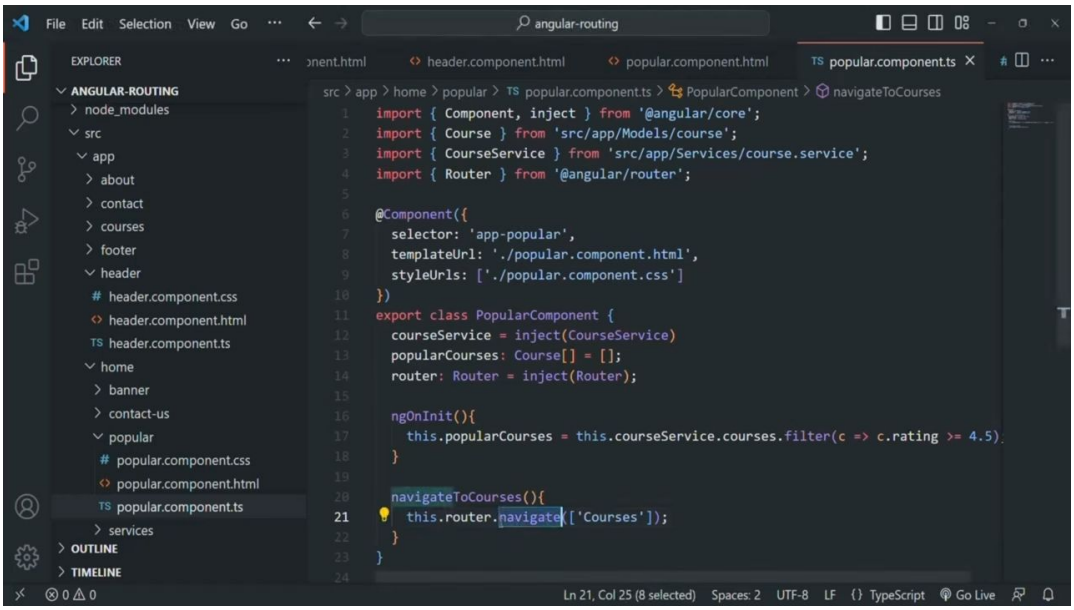
URL: `localhost:4200/Books/Author/101`

We can pass each segment of above URL as an element of the passed array to navigate method.

```
this._router.navigate(['Books', 'Author', 101]);
```

The navigate method on a router allows for programmatically navigating between routes using an array of route segments as arguments.

1:06:28



Successful redirection to the courses page using the navigate method on the router object has been achieved.

1:08:34

navigateByUrl() Method

Using **navigateByUrl()** method, we can navigate from one route to another programmatically. The **navigateByUrl** method takes a string value as an argument and that string value should contain all the route segments.

Let's say, we want to navigate the user to following route:

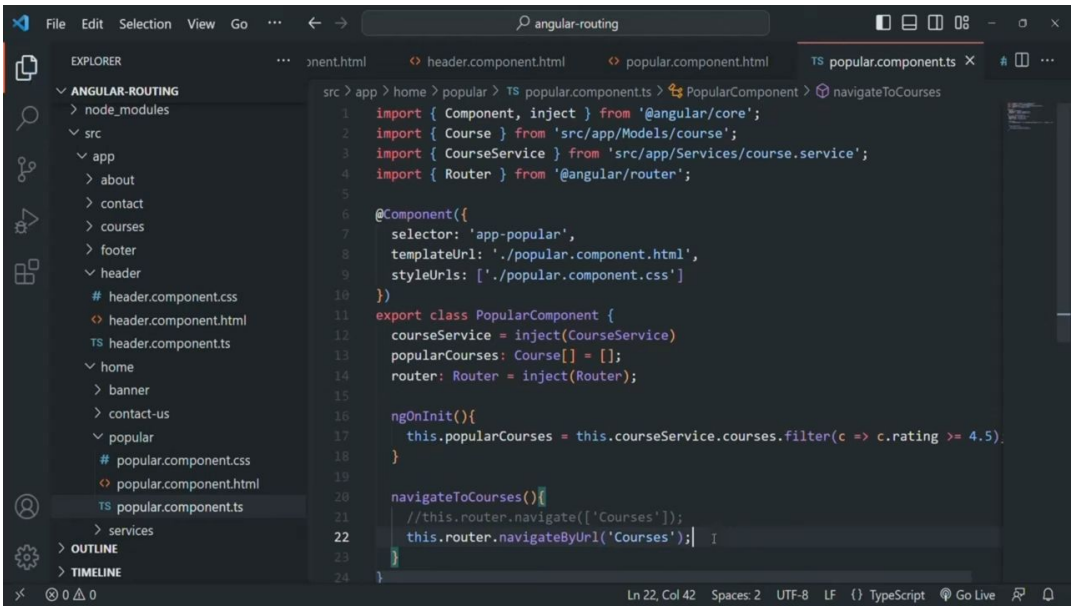
URL: `localhost:4200/Books/Author/101`

We can pass each segment of above URL as an element of the passed array to navigate method.

```
this._router.navigateByUrl('Books/Author/101');
```

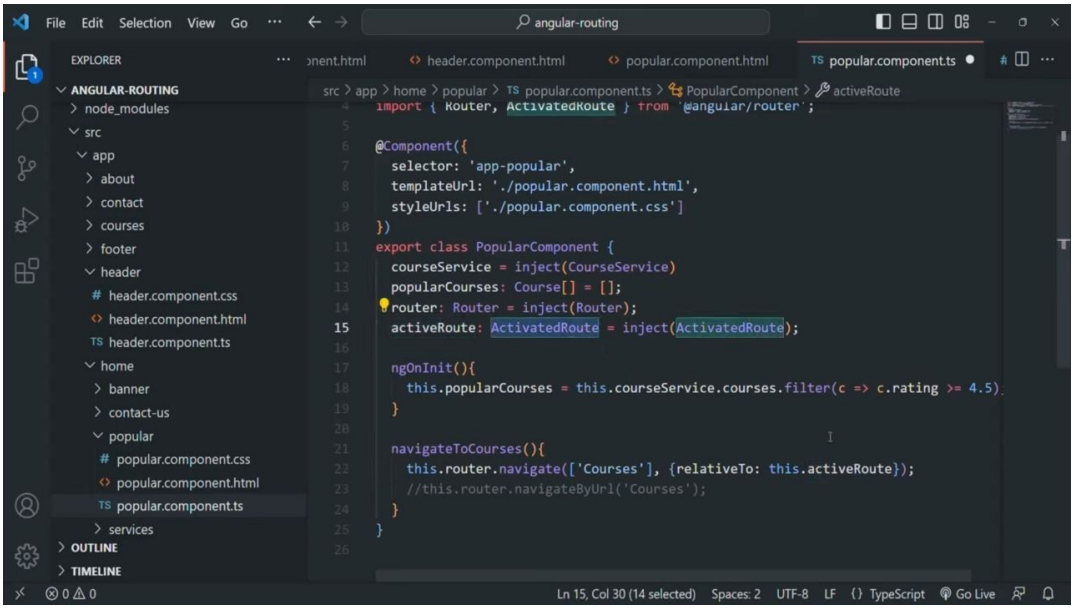
The navigate by URL method on the router object allows for programmatic navigation between routes using a string value containing all route segments.

1:08:58



The navigation method on the website successfully directs users to the courses page, providing a seamless user experience.

1:09:46

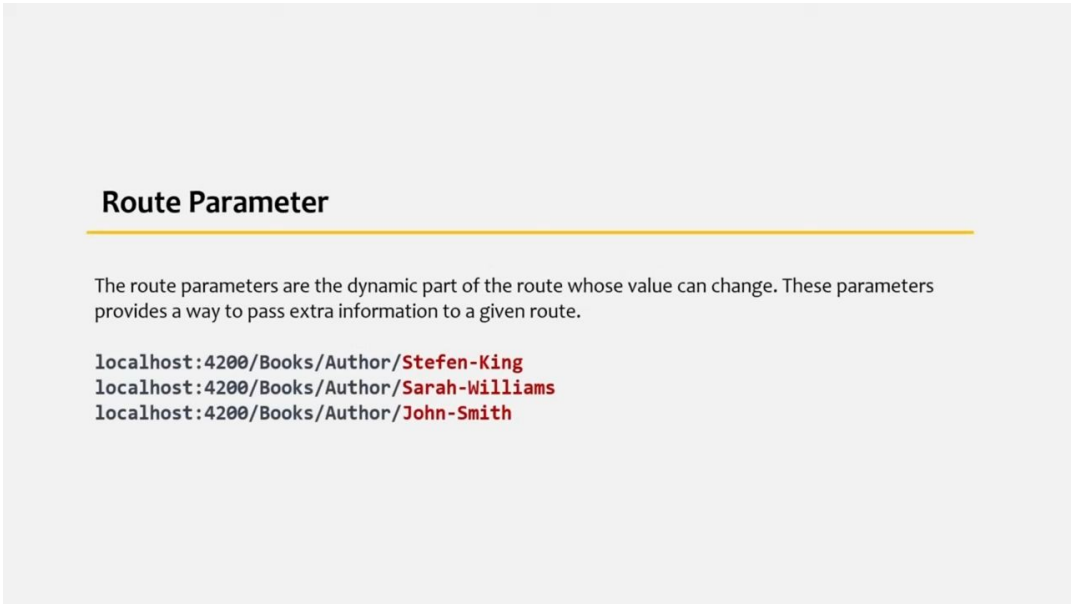


The active route in the navigate method stores information about the currently active route.

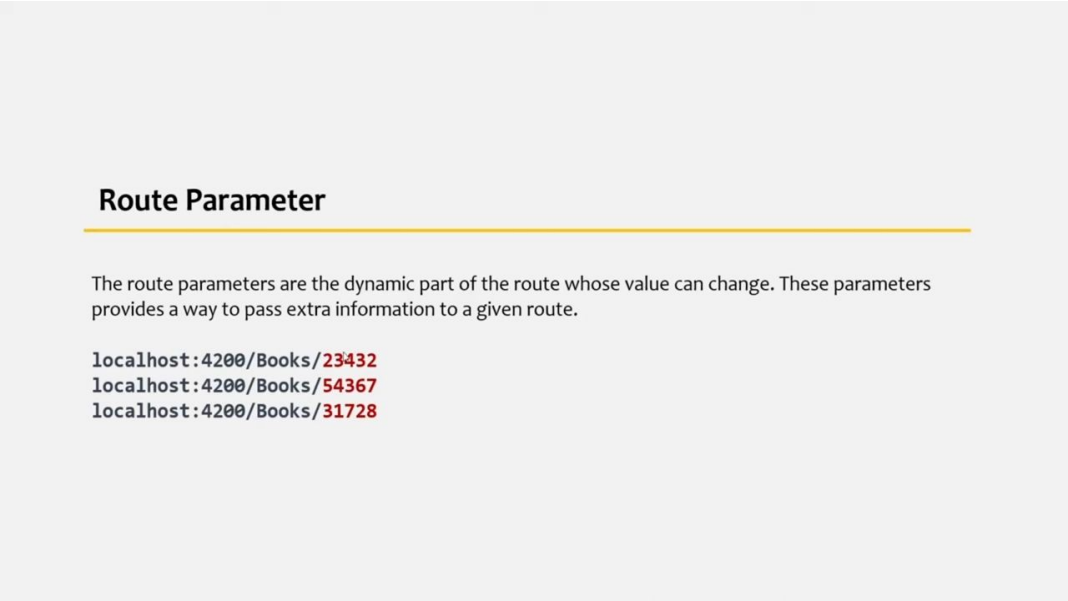
1:13:17

Angular route parameters allow for dynamic values to be passed to a given route, providing extra information and functionality.

1:14:56

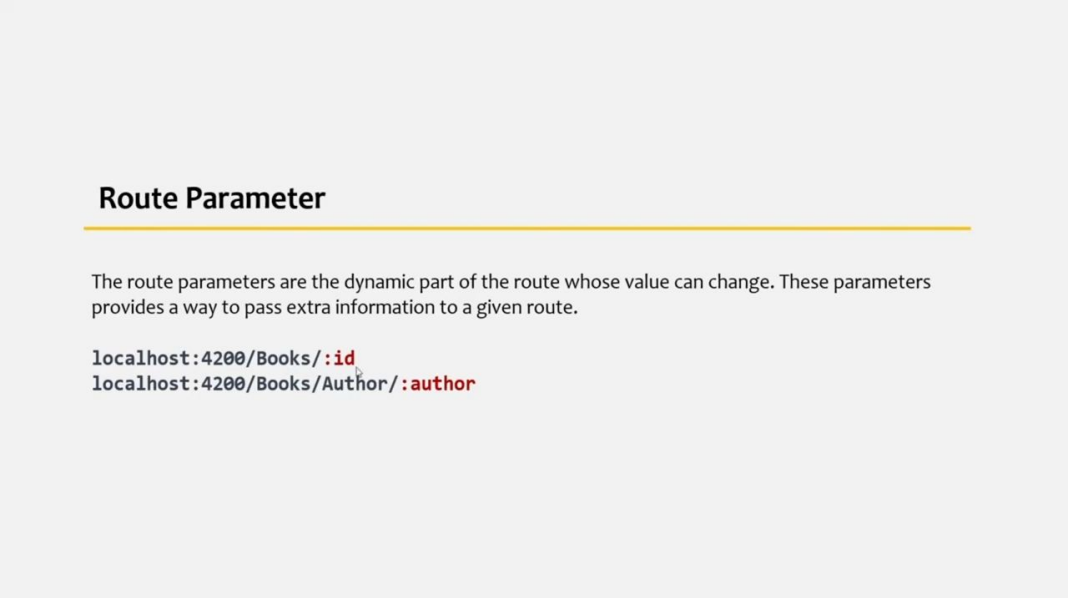


1:15:05



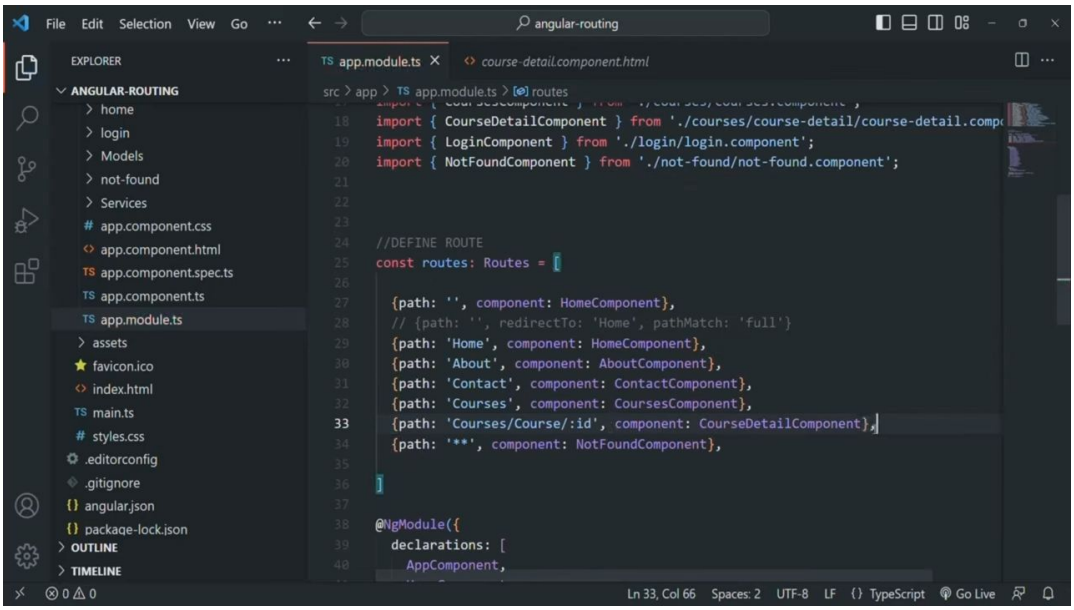
Using route parameters in Angular allows for dynamic changes to the URL path, providing flexibility and customization for web applications.

1:15:50



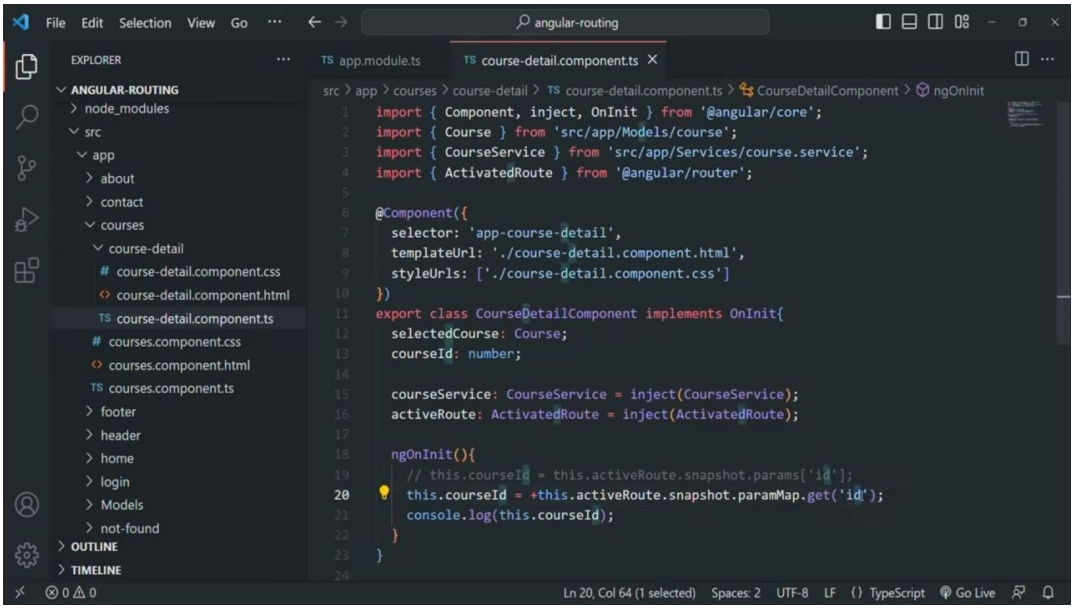
Angular allows for dynamic routing by using route parameters, which can be specified with a colon before the parameter name.

1:16:20



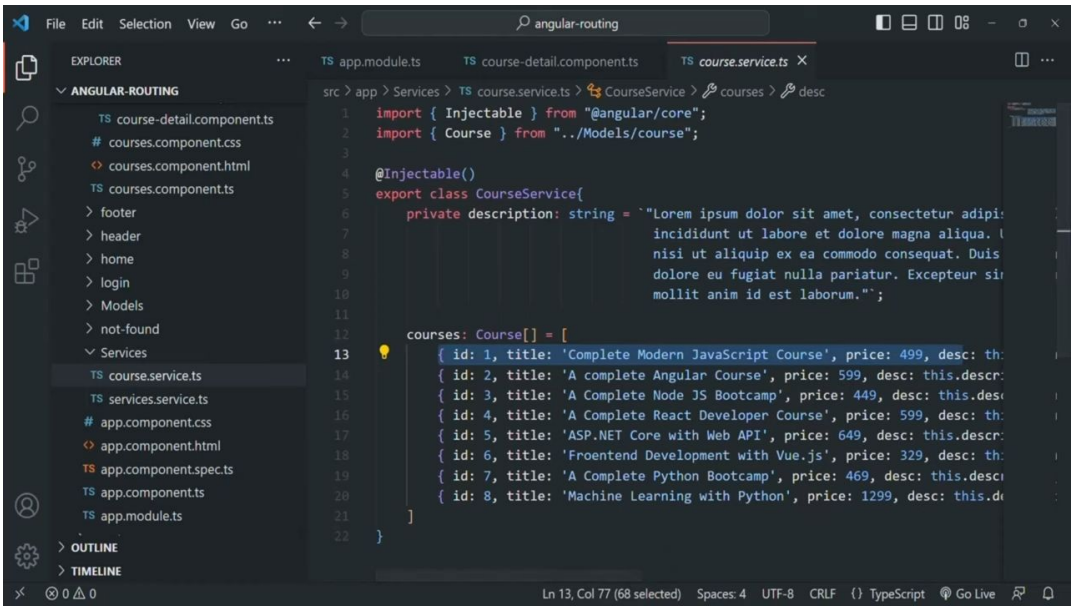
Using the component property, users can assign course detail components to specific URLs for easy access.

1:22:11



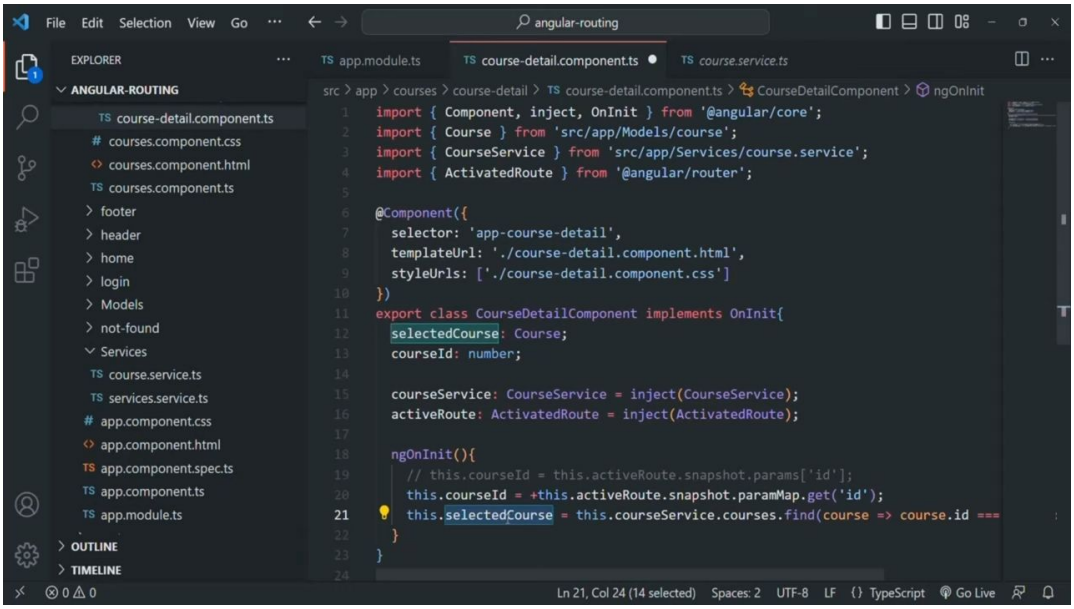
A new method of logging and filtering course values based on ID route parameters has been implemented in the courses service.

1:30:26



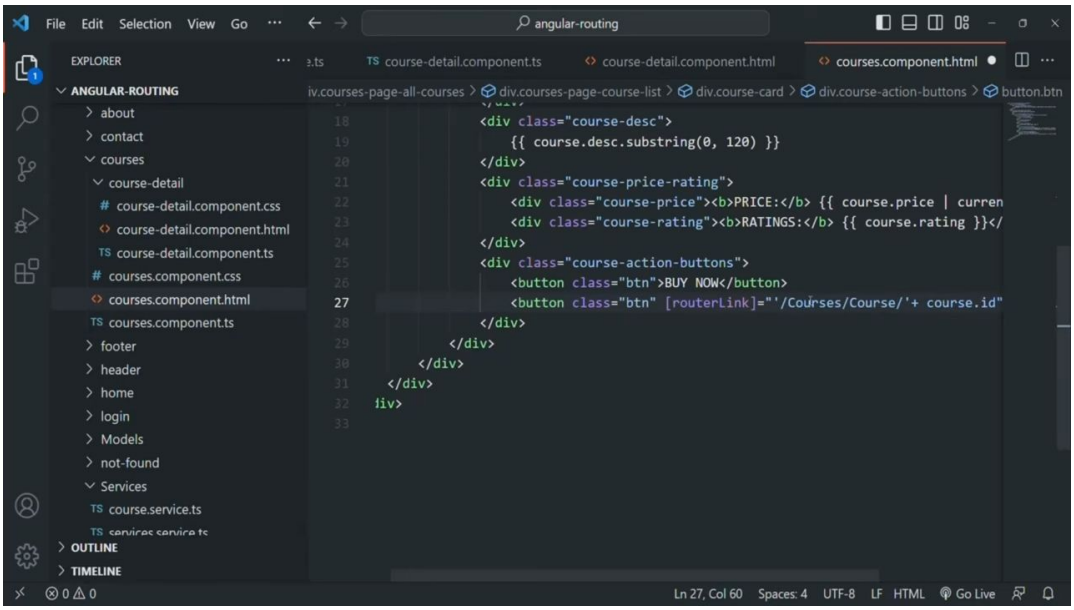
Filtering course details based on the ID route parameter is a crucial step in displaying specific course information in the course detail component.

1:30:57



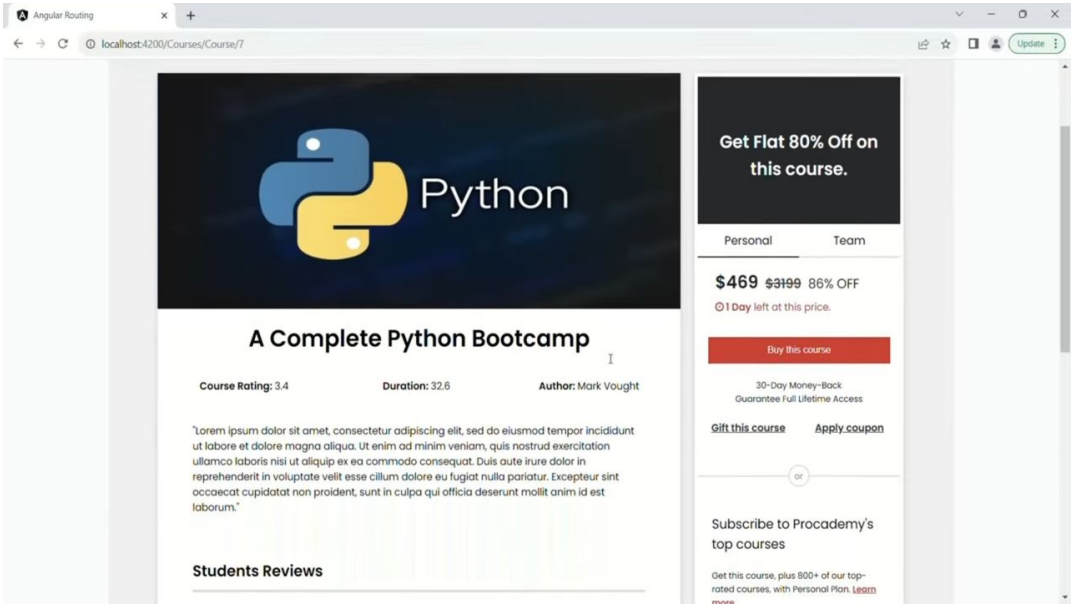
New method for displaying course details has been implemented in the course detail component.

1:33:18



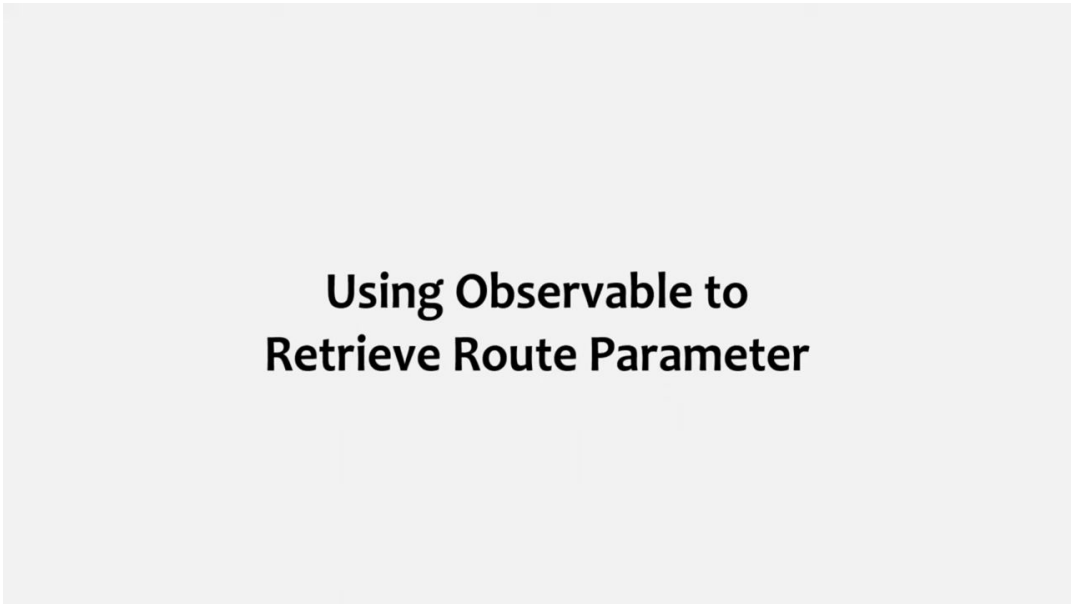
Using absolute paths instead of relative paths is crucial for ensuring that the correct URLs are accessed in web development.

1:37:33



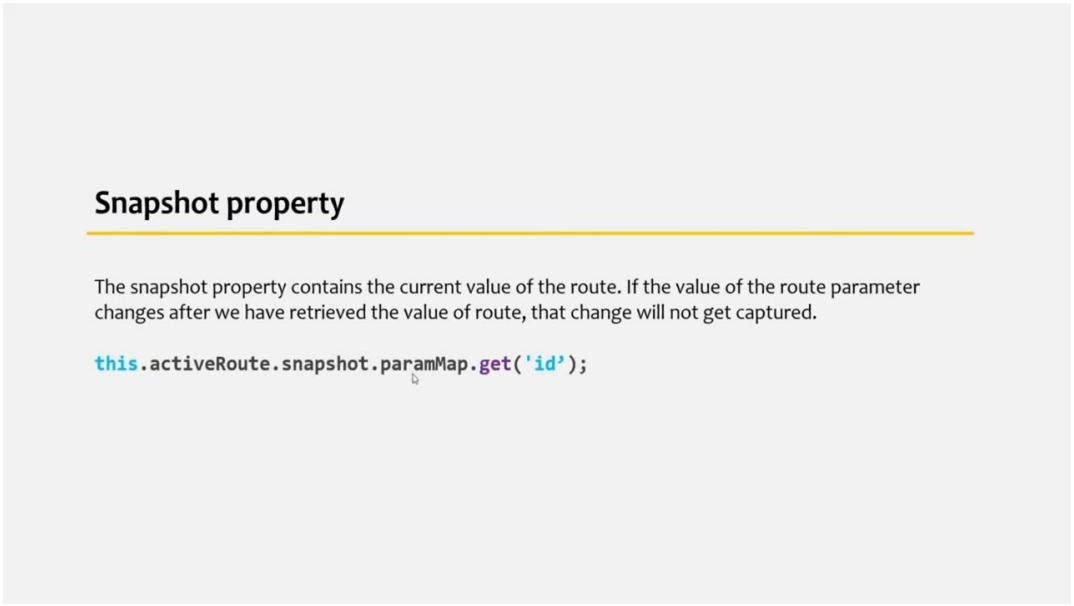
The new course details display feature in the app module allows for easy navigation and access to specific course information.

1:37:59

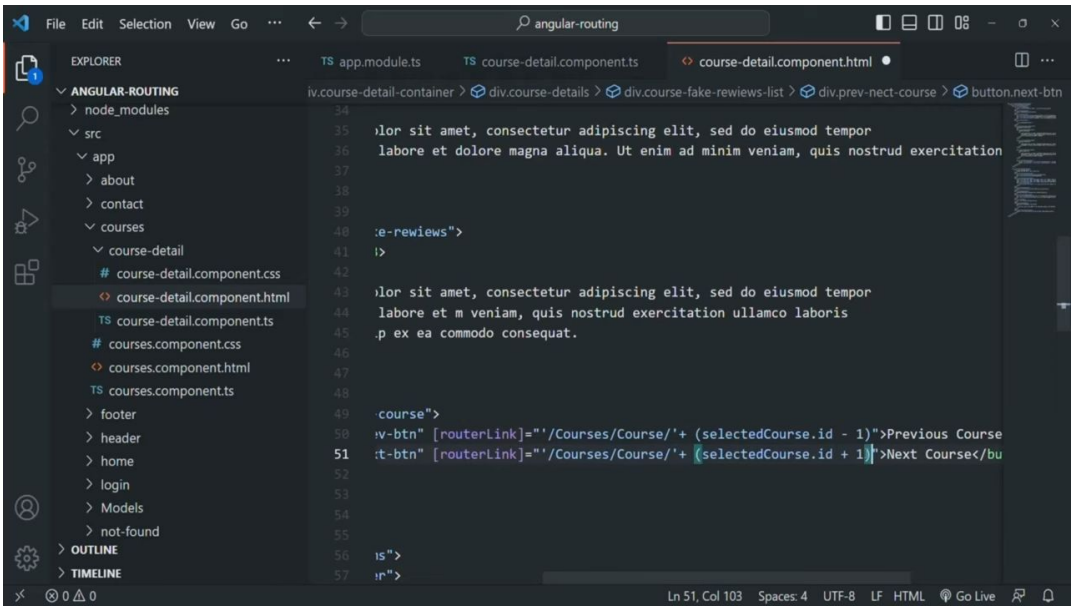


Learn how to dynamically display content based on route parameters in Angular.

1:39:29

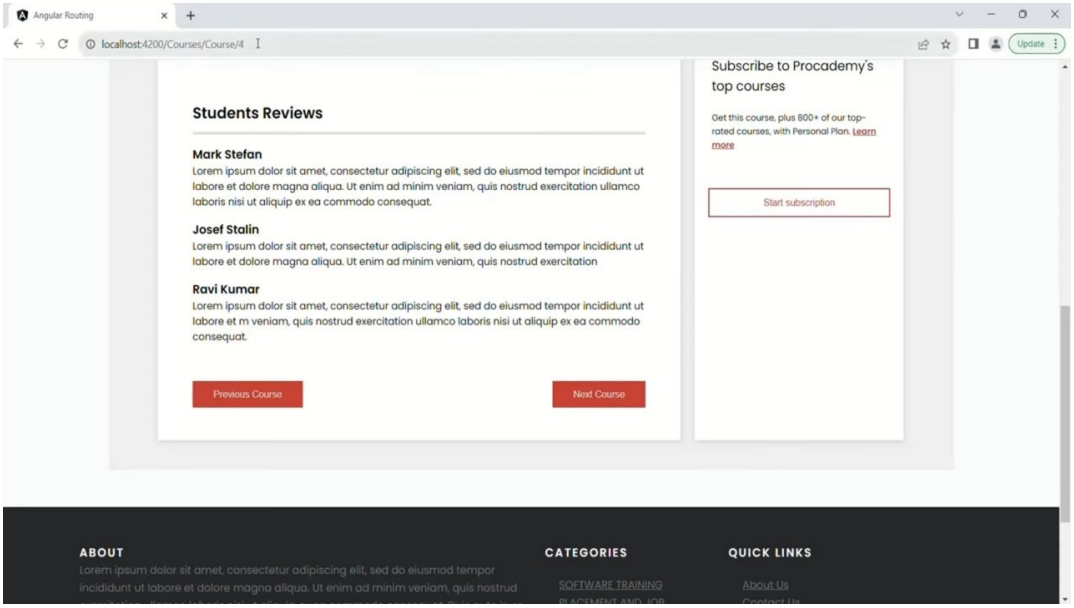


1:39:37



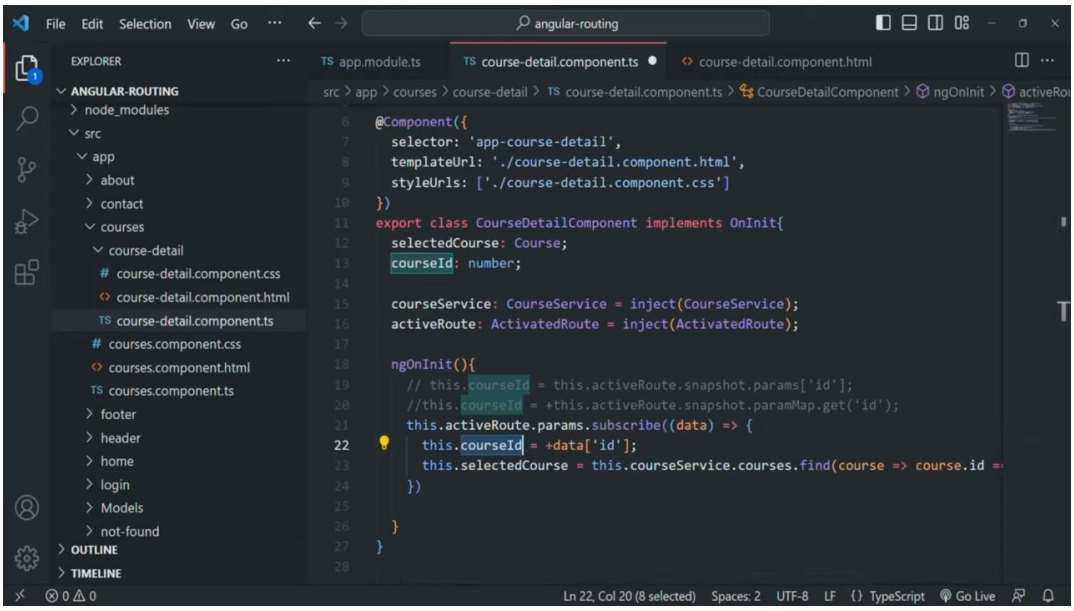
A programming error was fixed on the website, allowing users to access course details without any issues.

1:42:30



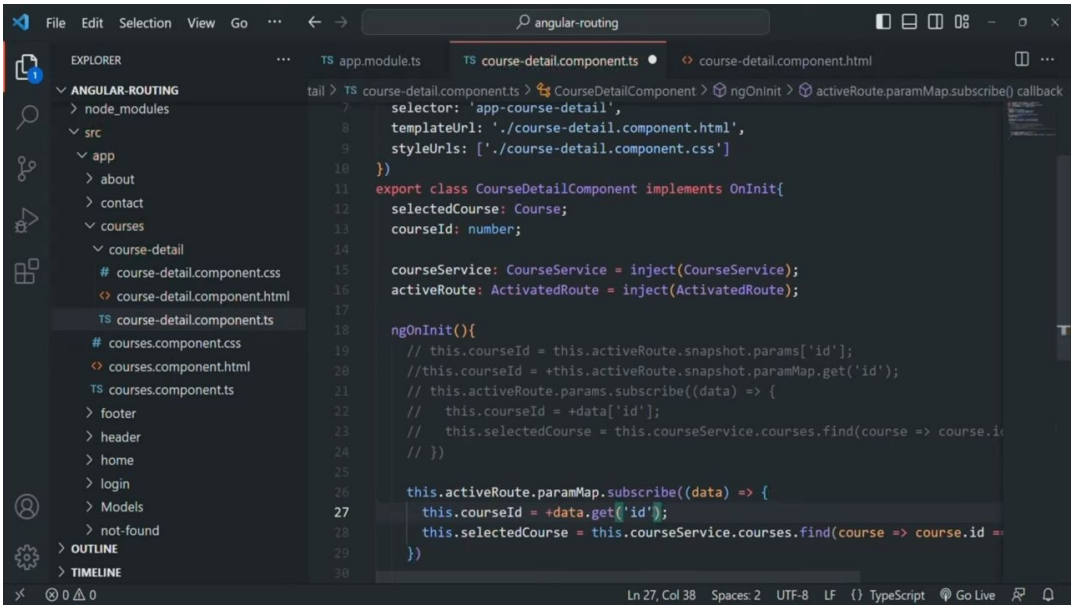
Course navigation on the new platform has been causing issues for users due to a glitch in the ID system.

1:43:10



New route values are being used to select courses based on the ID parameter in a web page.

1:49:16



Utilizing the get method, we can retrieve the value of a parameter from the currently active route in order to ensure seamless functionality on web pages.

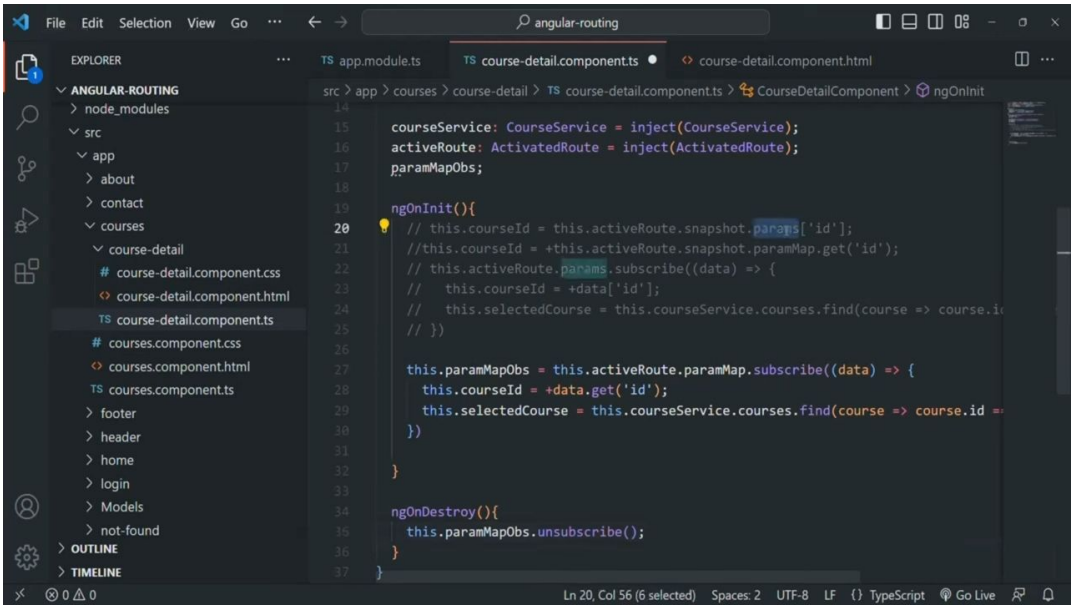
1:50:58

The paramMap Observable

By subscribing to the paramMap observable (or to params observable), you will get a notification when the value changes. Hence, you can retrieve the latest value of the parameter and update the component accordingly.

Subscribing to route parameter observables allows for real-time updates and ensures proper component functionality.

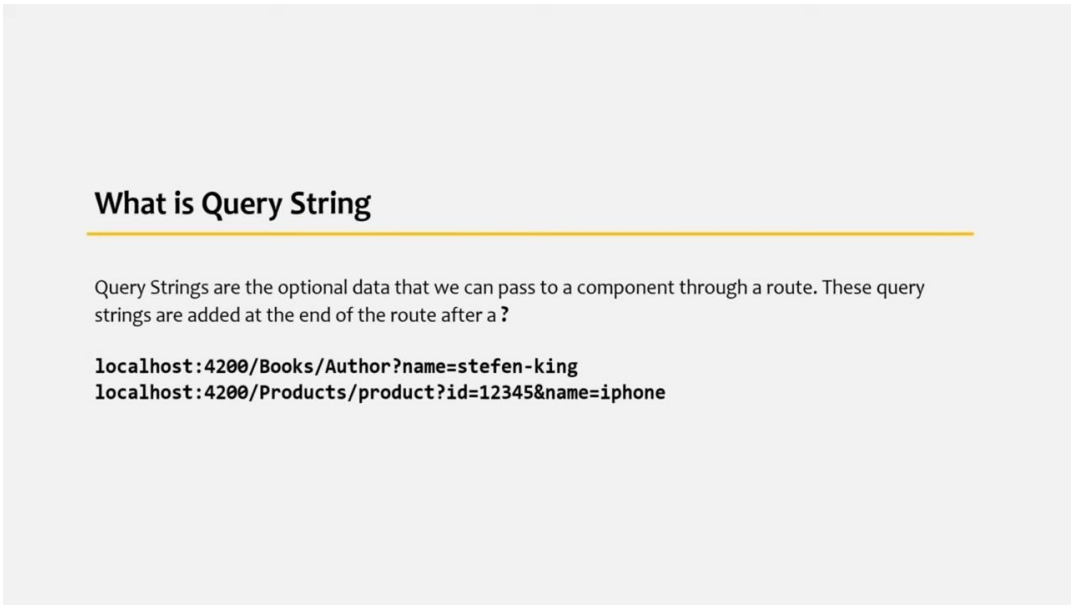
1:51:49



1:53:51

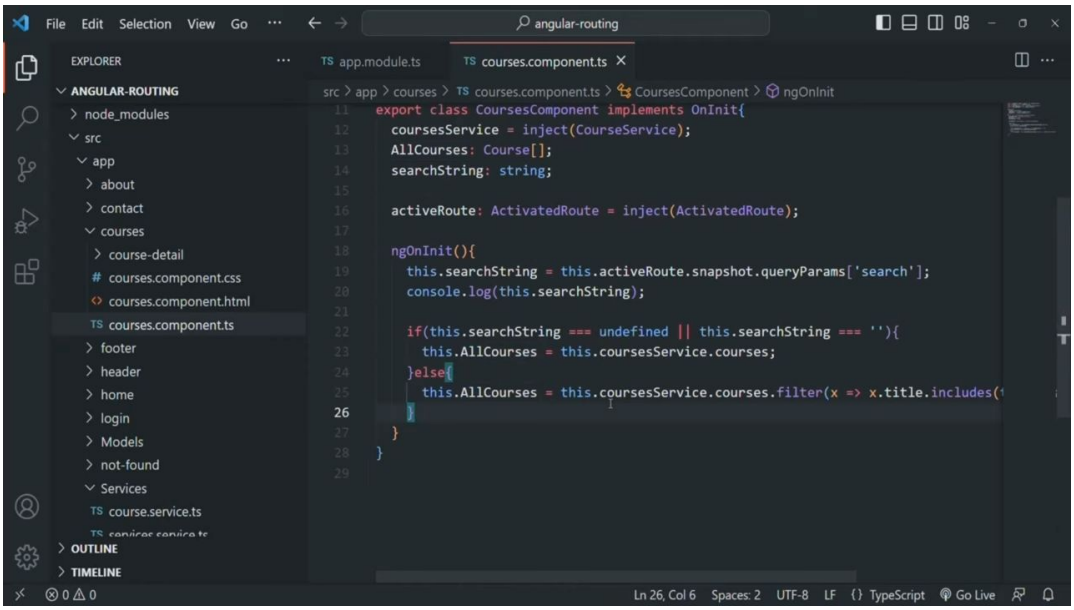
The use of the params observable is deprecated in Angular, and it is recommended to use the param map property instead for reading Route parameters.

1:54:03



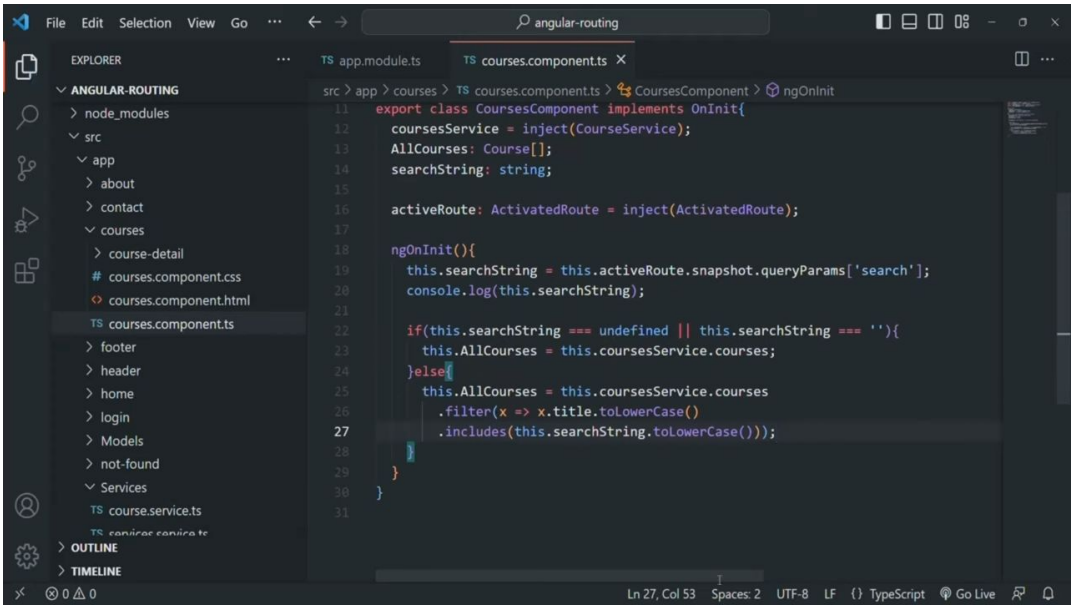
Learn how to specify and retrieve query string values in Angular routes for passing optional data to components.

1:54:19



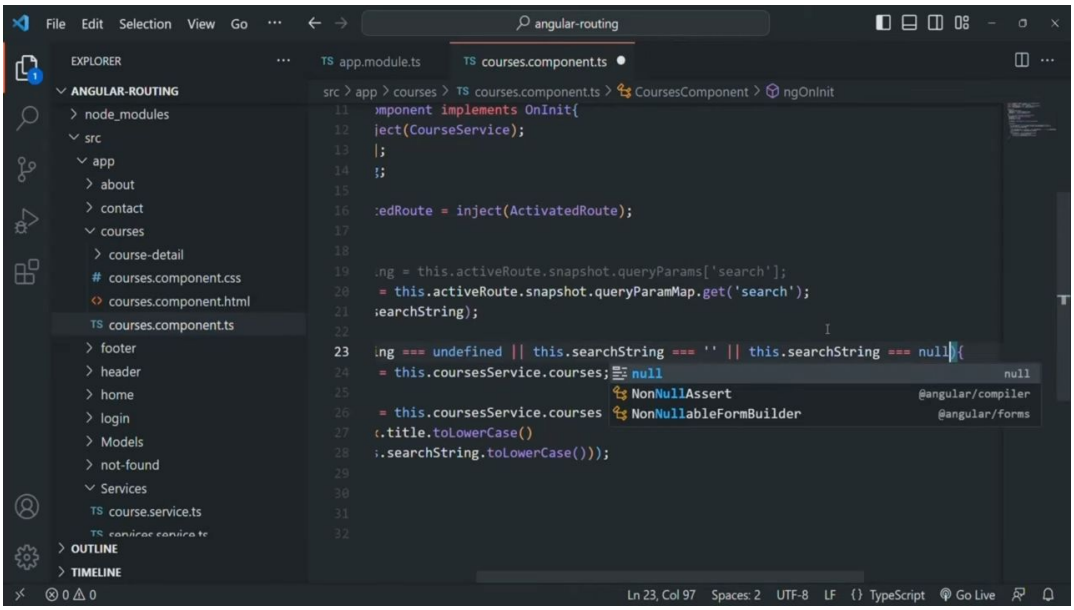
Filter method used to search for courses with the term "JavaScript" returned no results.

2:03:24



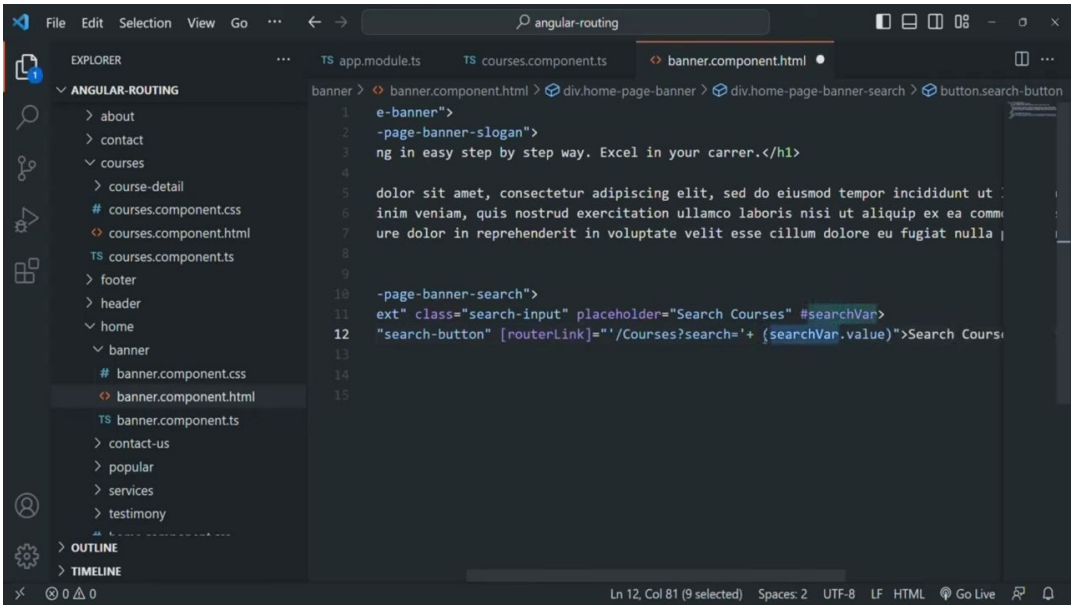
Search strings are converted to lowercase before comparison, allowing for more accurate and visible results on the web page.

2:04:12



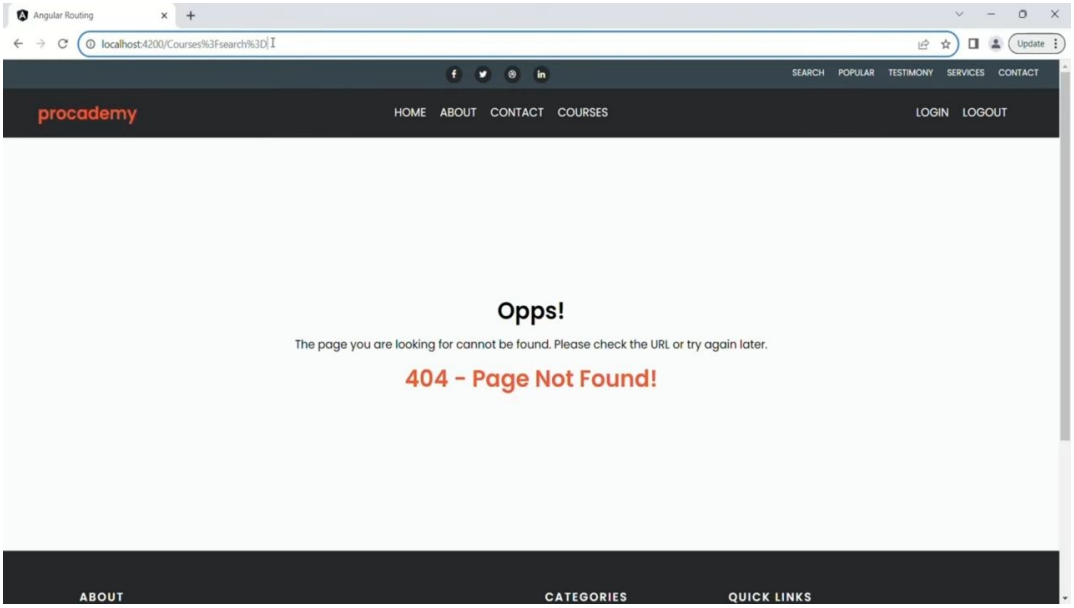
New feature added to web page allows for improved course search functionality.

2:06:14



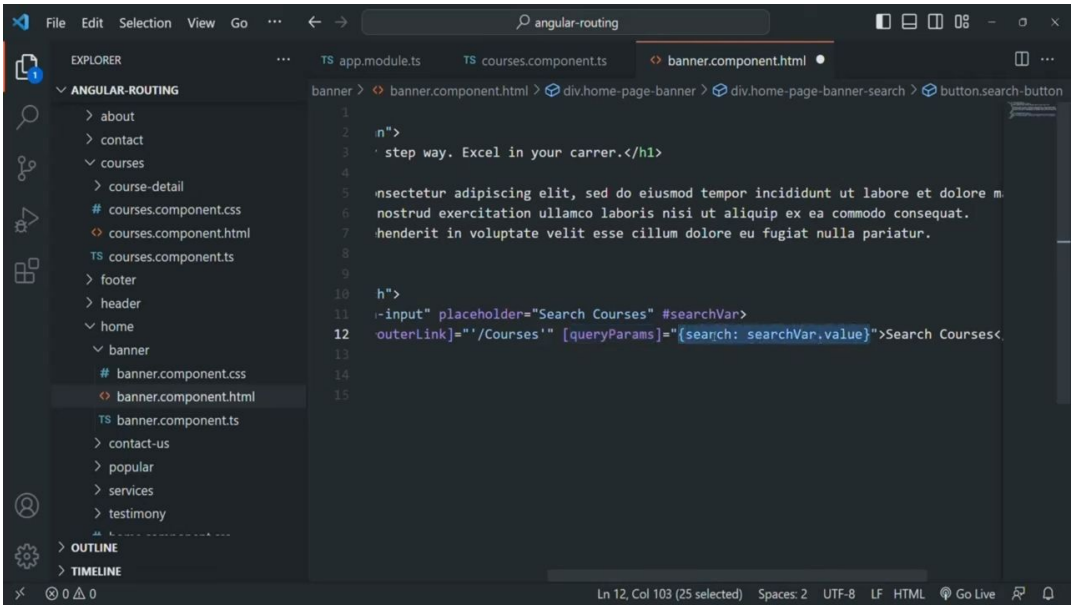
Template reference variables in JavaScript can be used to directly access input element properties.

2:09:35



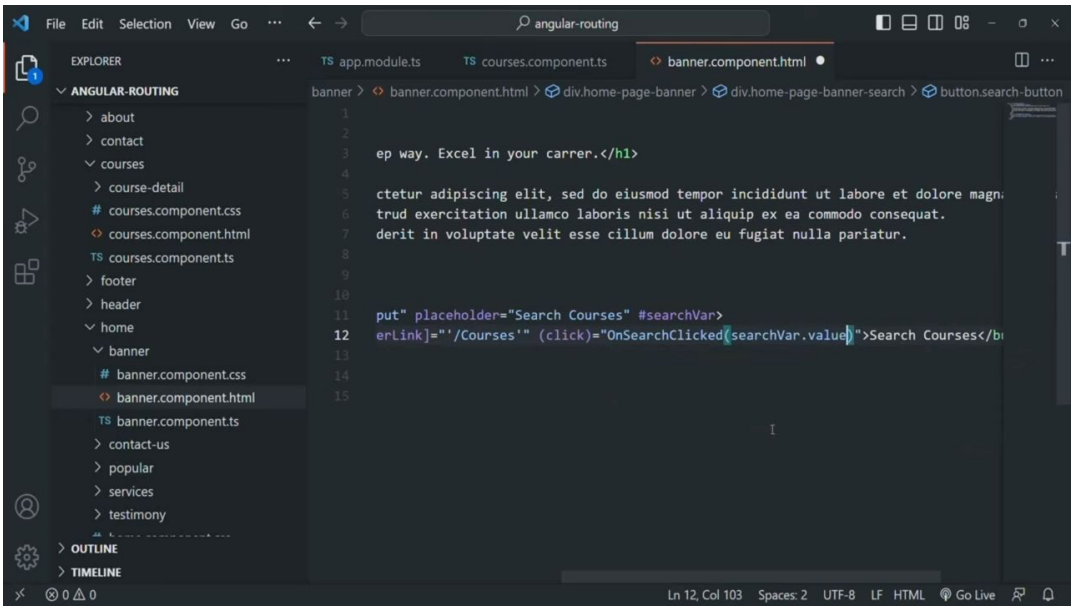
Website search function not displaying correct results due to improper URL formatting, leading to 404 error.

2:10:17



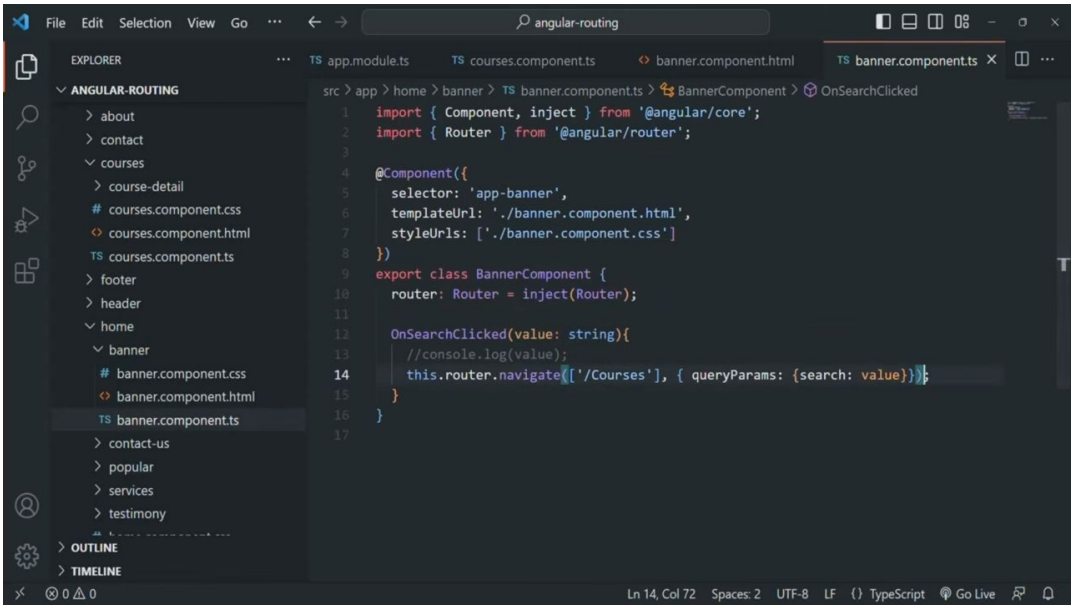
In order to specify query parameters, the query params directive is used to assign an object with the query string name and its value.

2:11:41



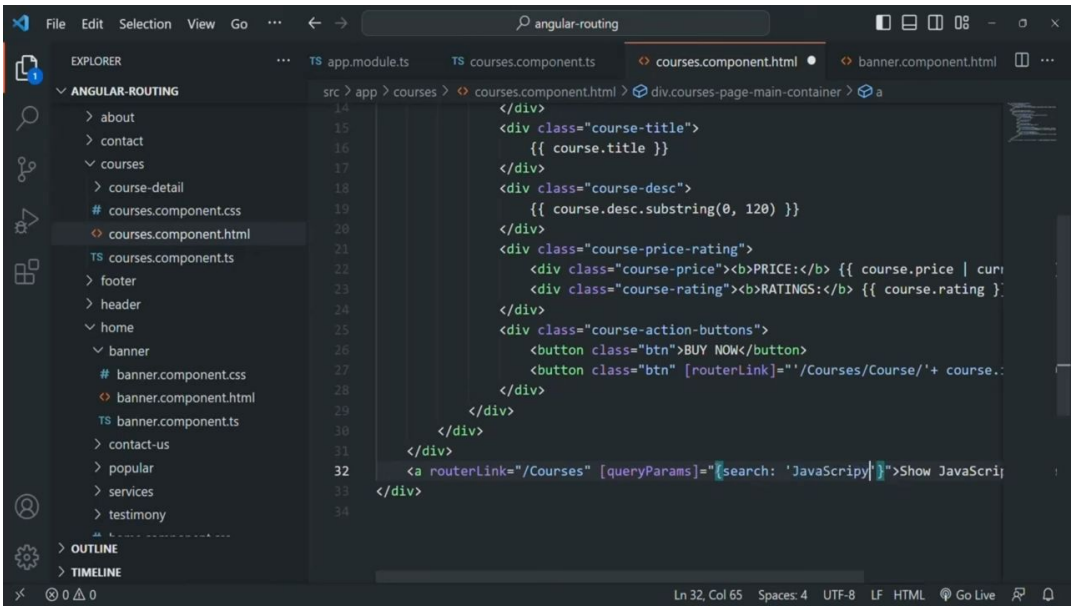
The Banner component in the search input element allows users to enter a value and trigger a search click method.

2:13:58



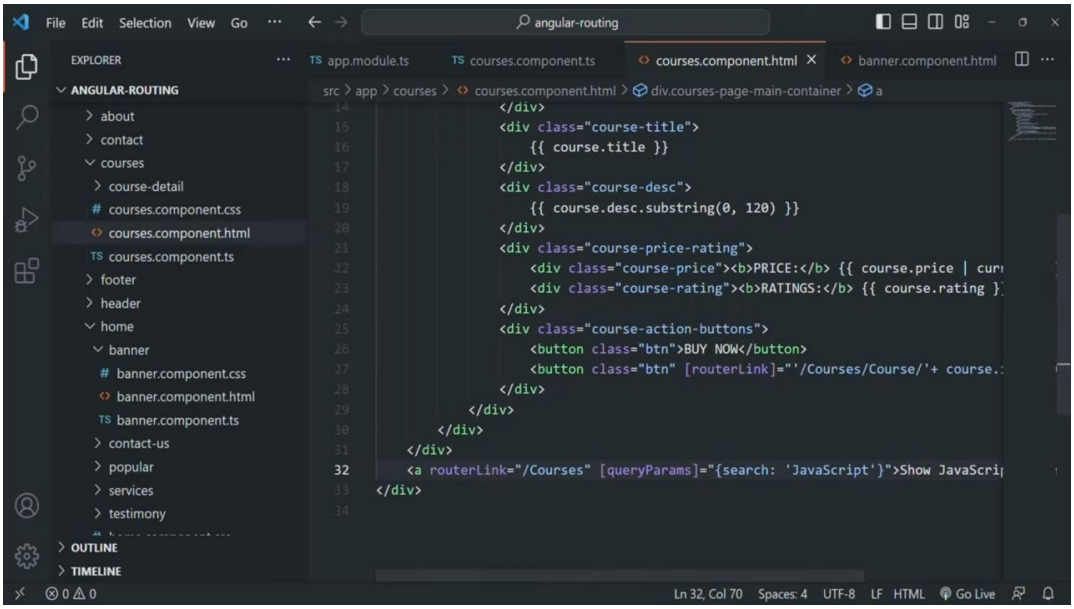
The ability to specify query strings in an object allows for efficient filtering of search results on web pages.

2:16:47

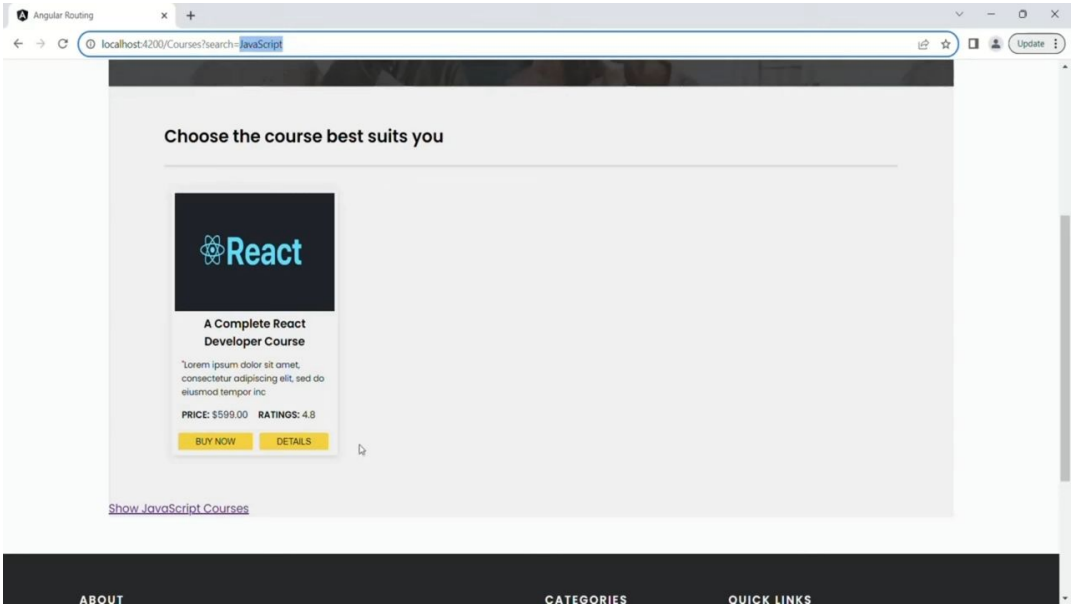


New feature on website allows users to easily search for JavaScript courses with a simple click.

2:19:16

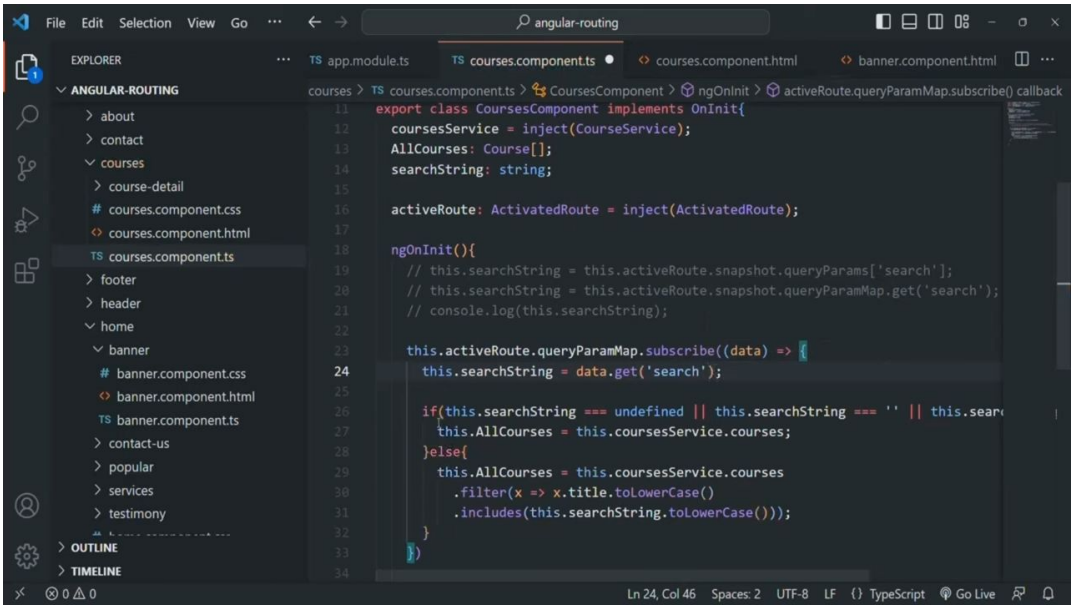


2:19:23



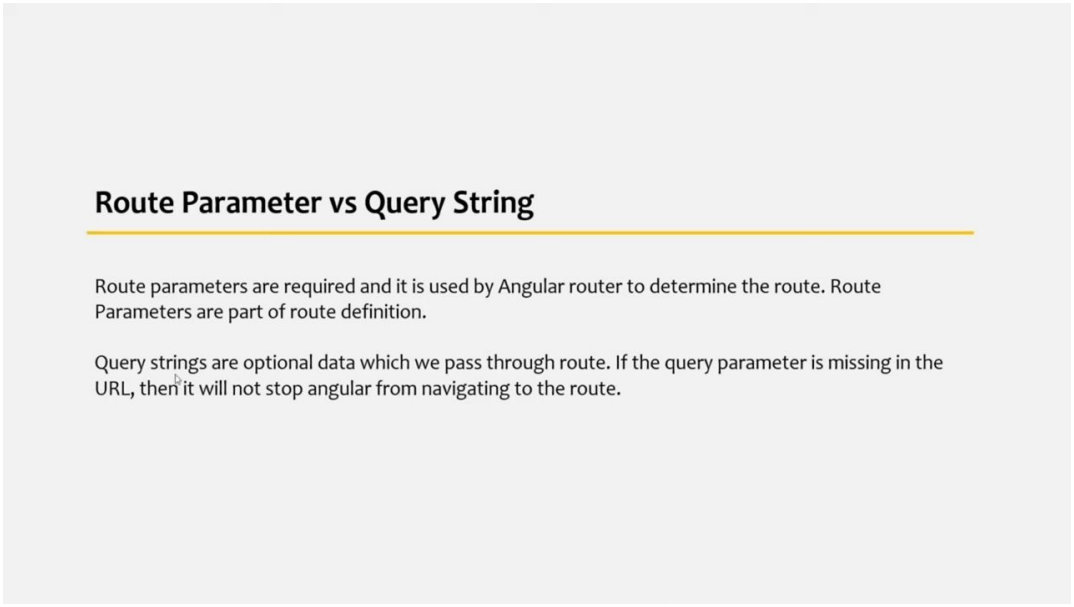
JavaScript course does not display as expected due to the logic being written inside the NG on init lifecycle hook.

2:20:07



Using the query param map observable, users can retrieve specific values from a map using the get method.

2:24:18



Route parameters are required for displaying specific results, while query parameters are optional data passed through a route.

2:26:59

The fragment in a route is a link that jumps to a specific section or content within an HTML page.

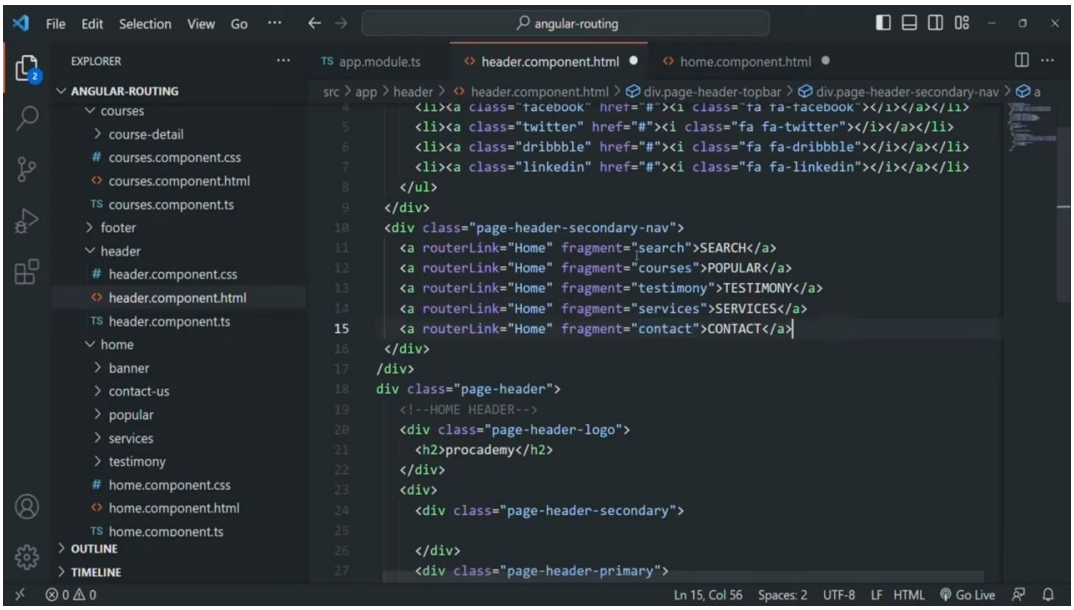
2:27:16

What is Query String

A fragment in a route is a link which jumps to a section or a content in the HTML page, which contains the ID mentioned in the fragment. A fragment comes after a # sign

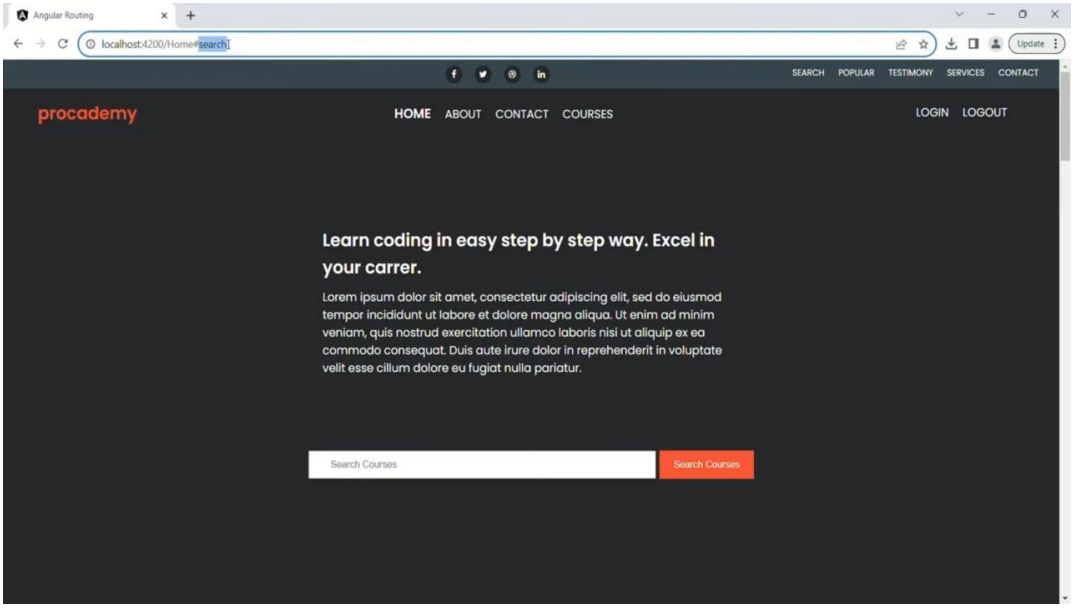
localhost:4200/Home#Services

2:27:27



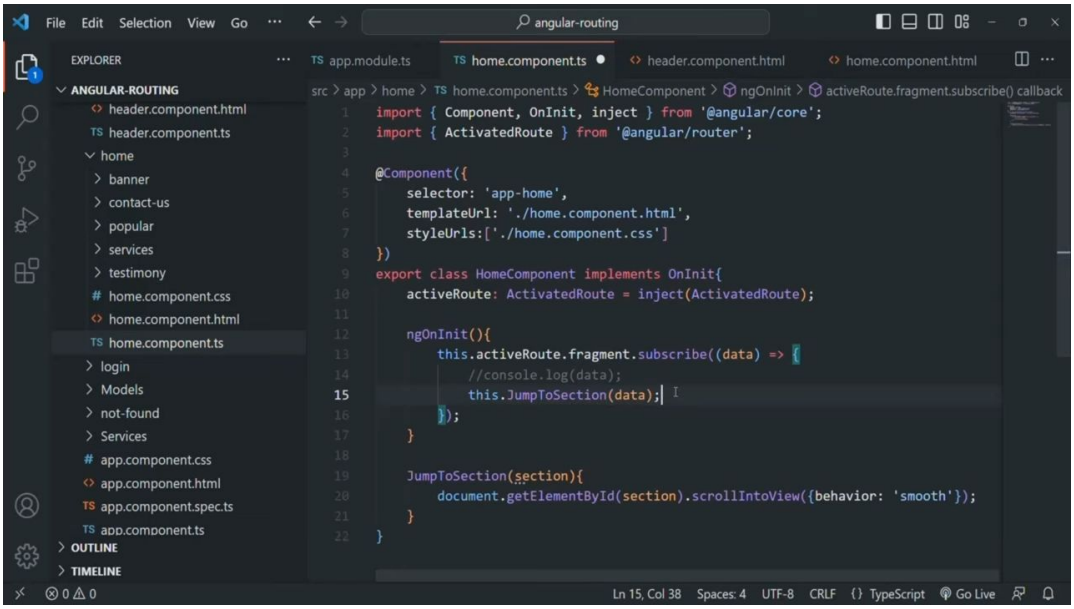
Developing a web page with specified fragments for different sections is the first step in creating a seamless user experience.

2:32:58



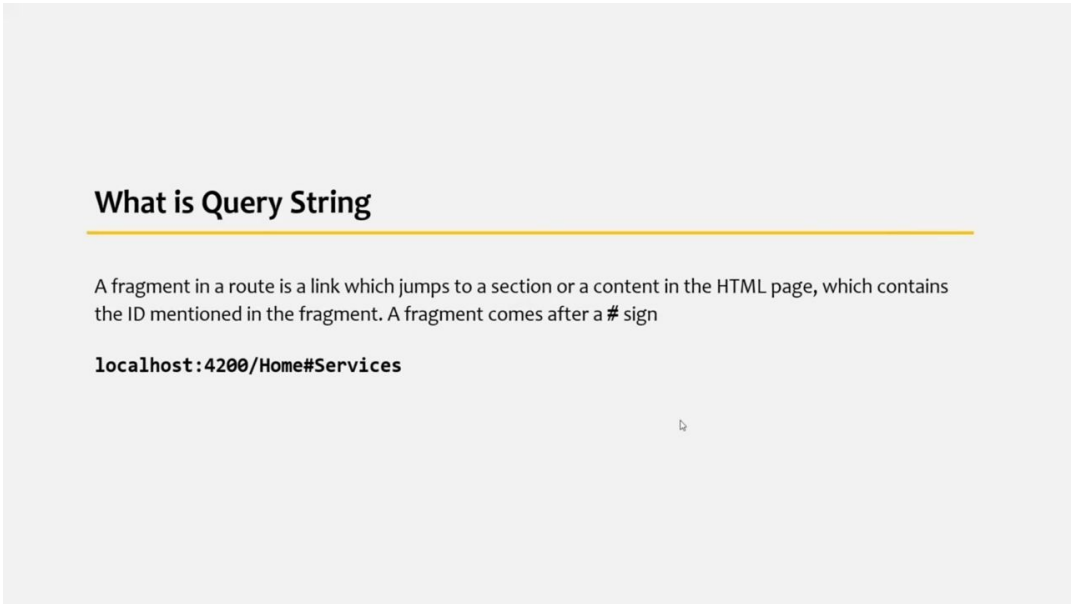
Web page navigation using fragment values is demonstrated by clicking on different links and observing the changes in the URL.

2:33:17



New method "jump to section" implemented for smoother navigation on web pages.

2:40:32

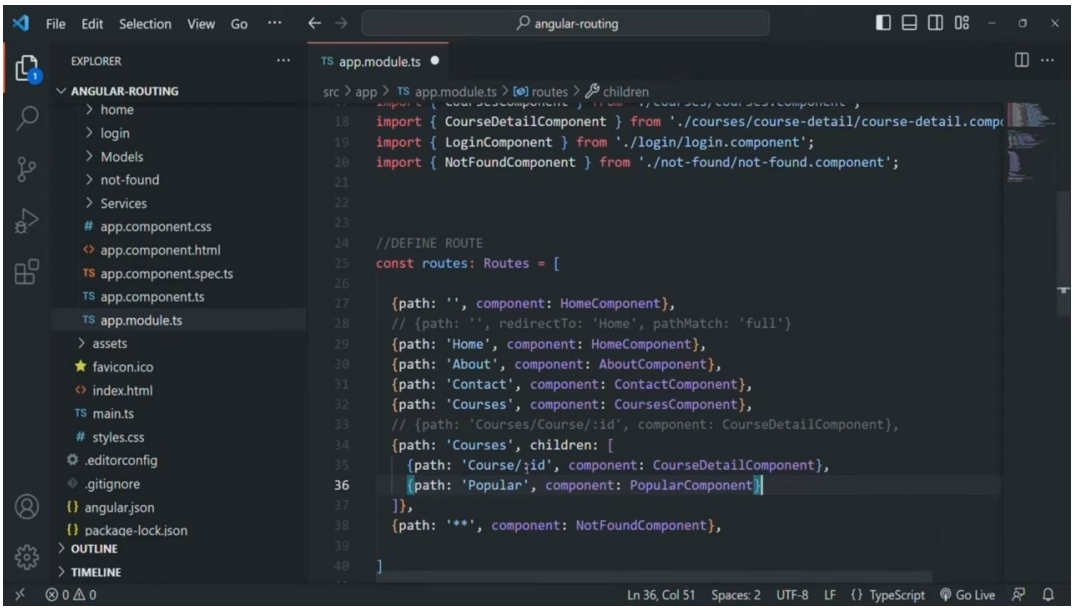


Utilizing fragments in HTML pages allows for easy navigation to specific sections of content.

2:41:29

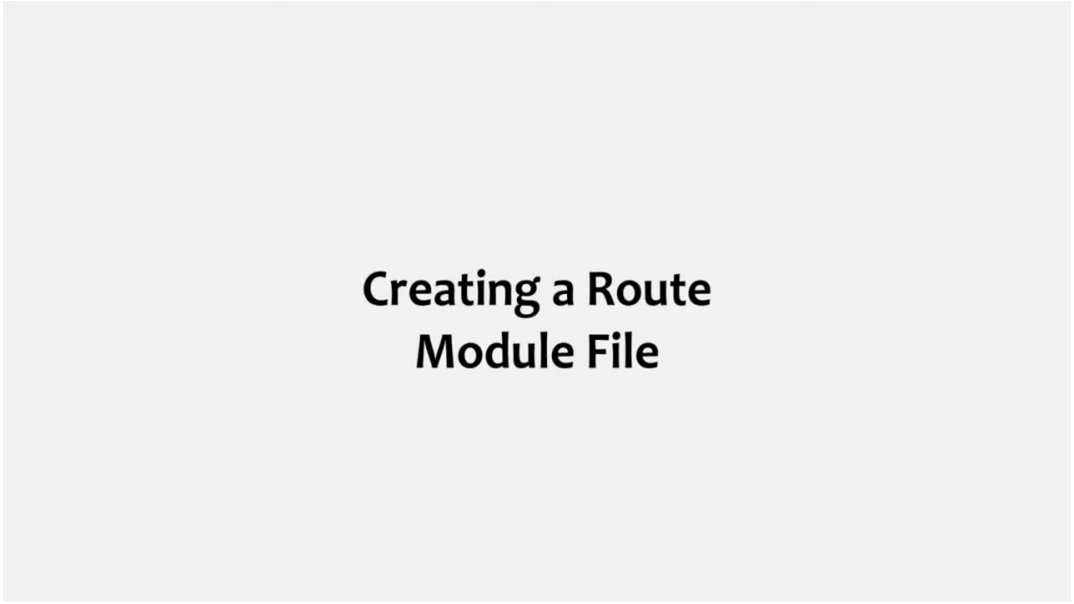
In this lecture, we will learn how to create and use a child route in Angular.

2:41:50



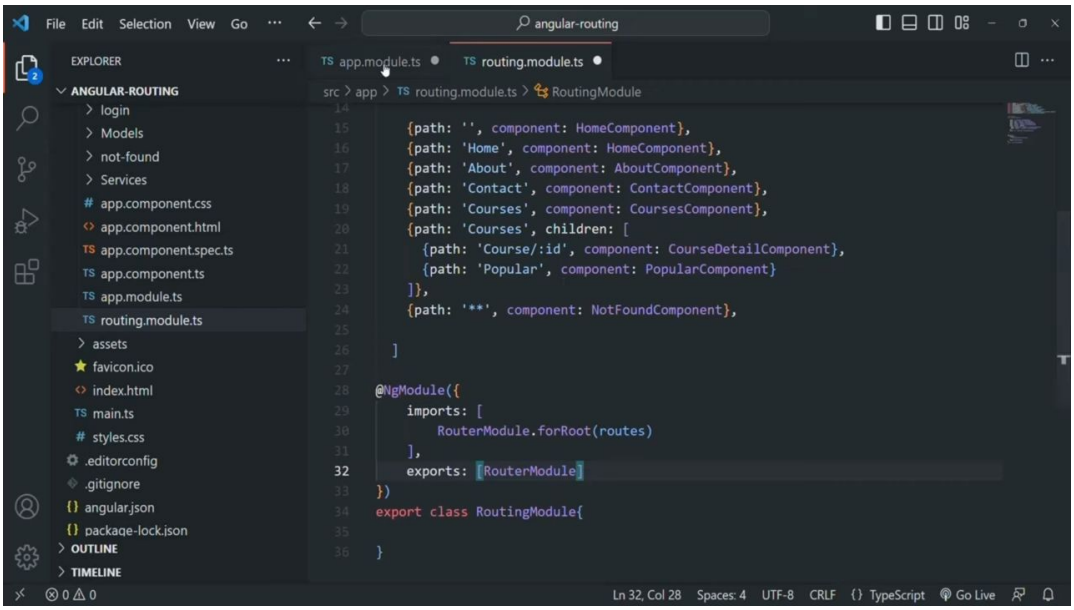
New Popular component added with two child routes for enhanced user experience.

2:46:43



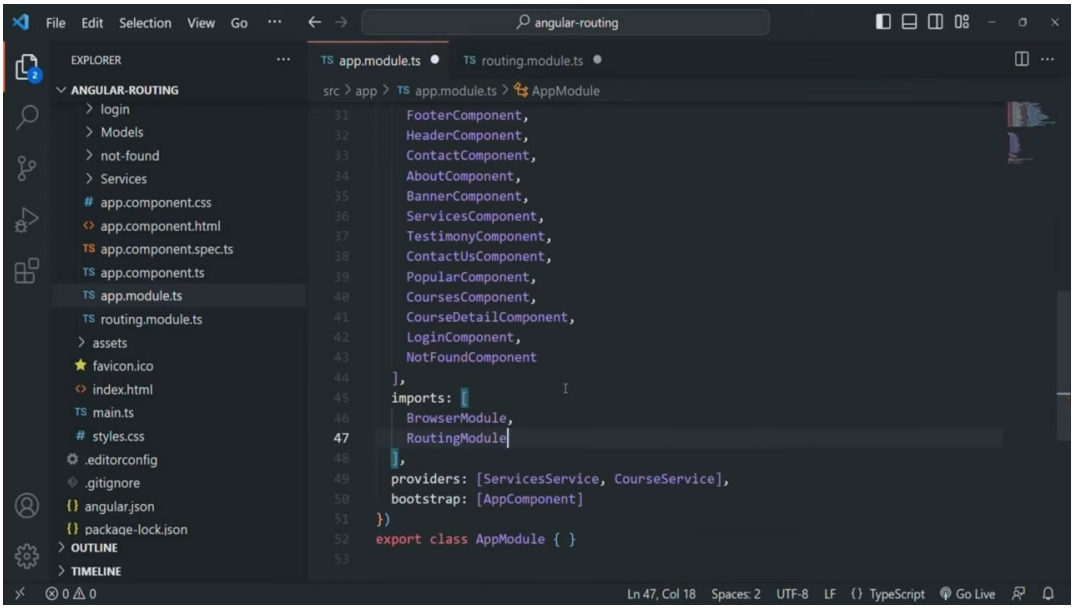
In the first lecture of this section, we learned how to define child routes in an Angular application.

2:47:27



Exporting and importing router modules allows for seamless integration of routing functionality across different files and modules in a web application.

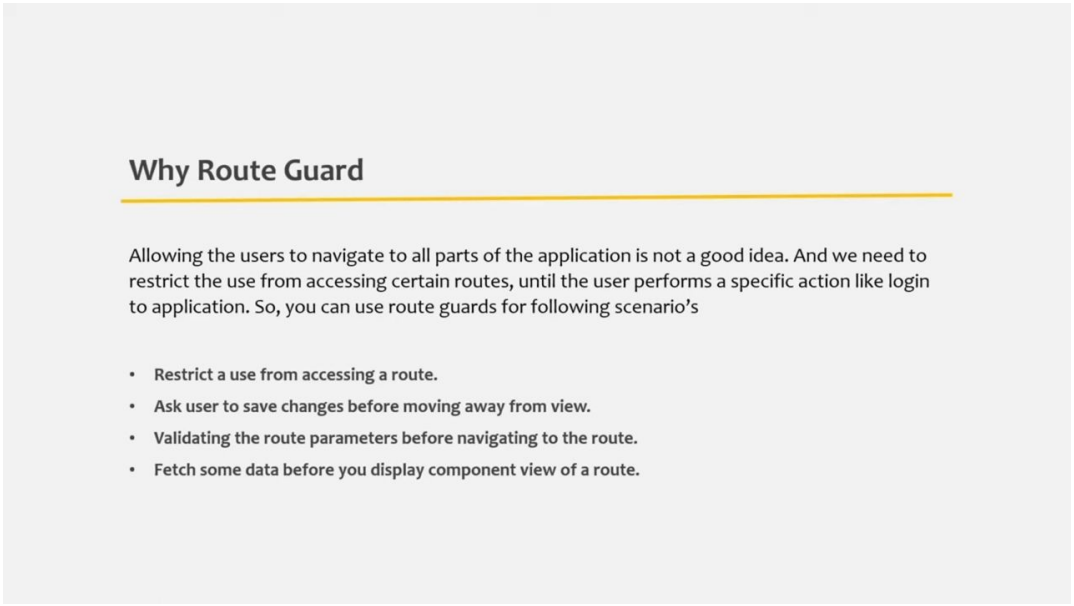
2:54:40



2:54:52

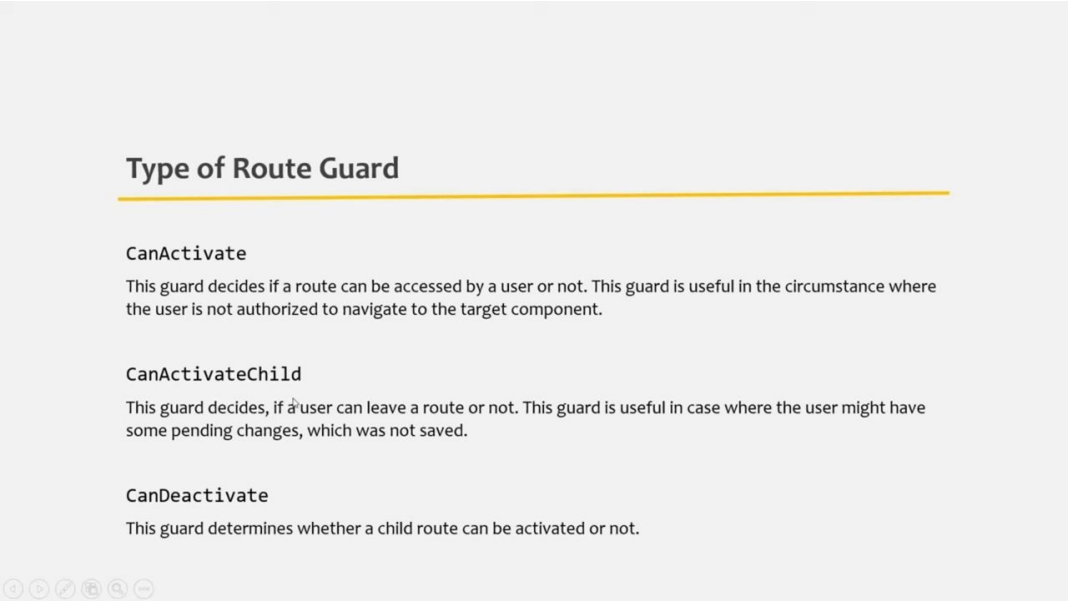
Route protection is essential for controlling access to certain parts of an application.

2:57:11



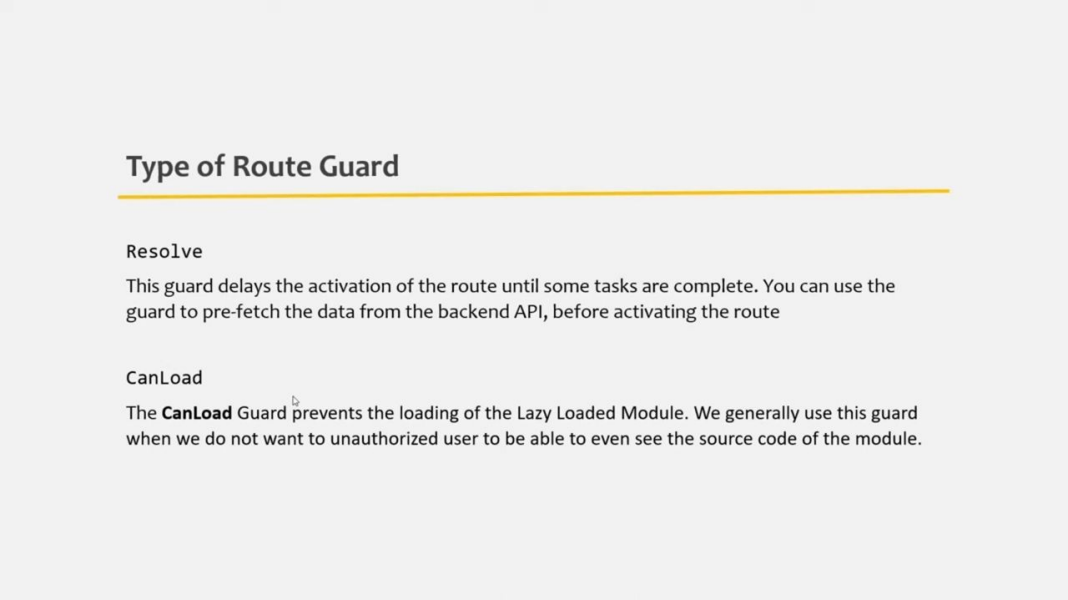
Angular route guards are a crucial tool for securing and controlling access to routes in web applications.

2:58:15



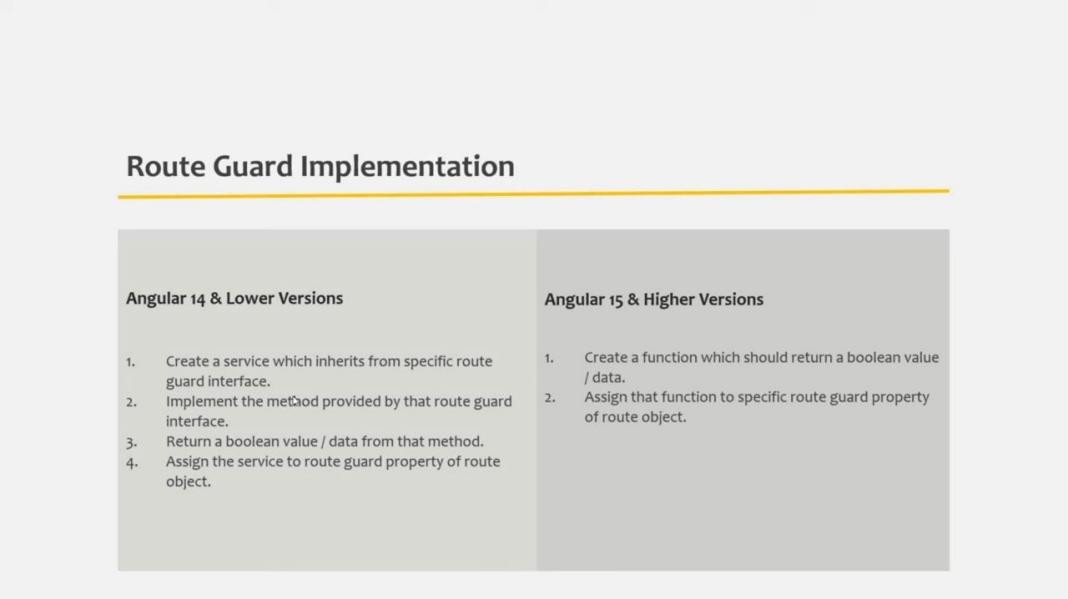
Angular introduces five route guards, including the connectivate and can activate child route guard, to control user access to routes.

2:58:56



New route guards in navigation systems allow for better control over user navigation and data prefetching.

3:00:42

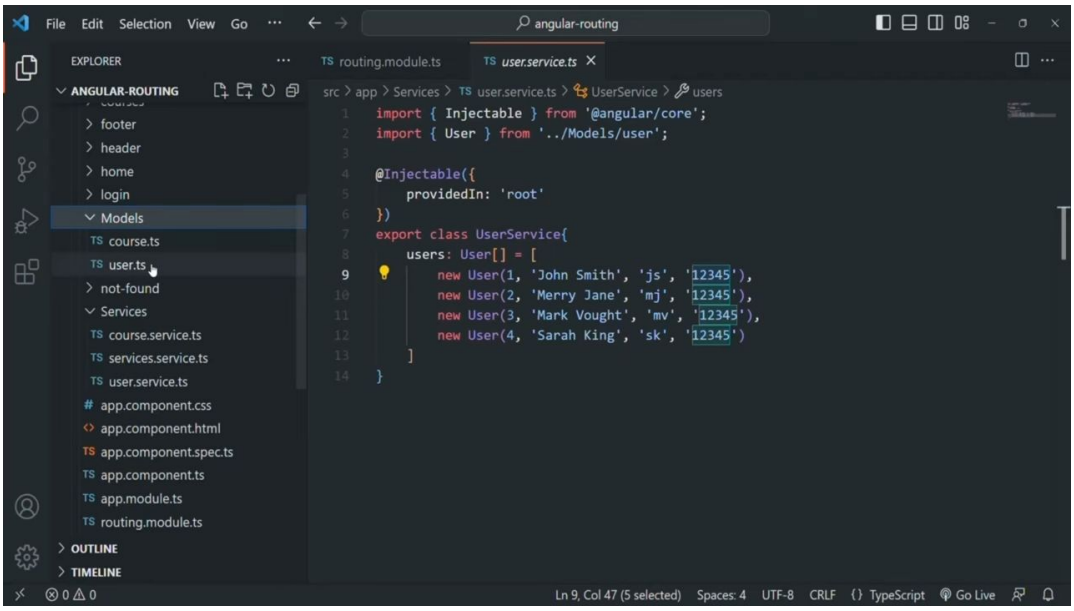


Angular has introduced a new way of implementing route guards in versions 15 and higher.

3:02:08

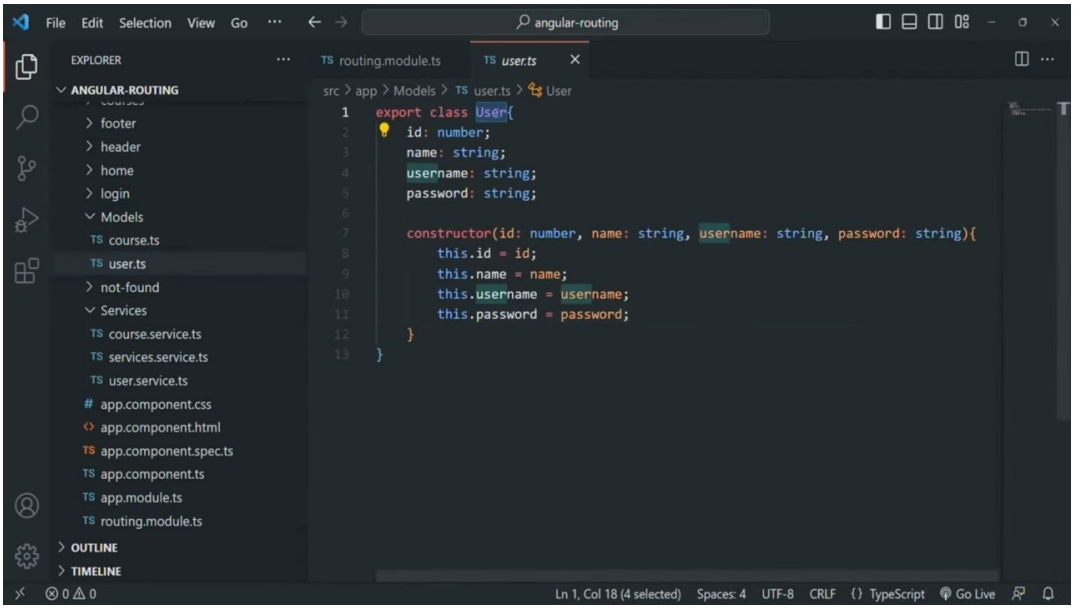
In this lecture, we will explore the implementation of route guards in Angular 14 and lower versions as well as in Angular 15 and higher versions.

3:04:47

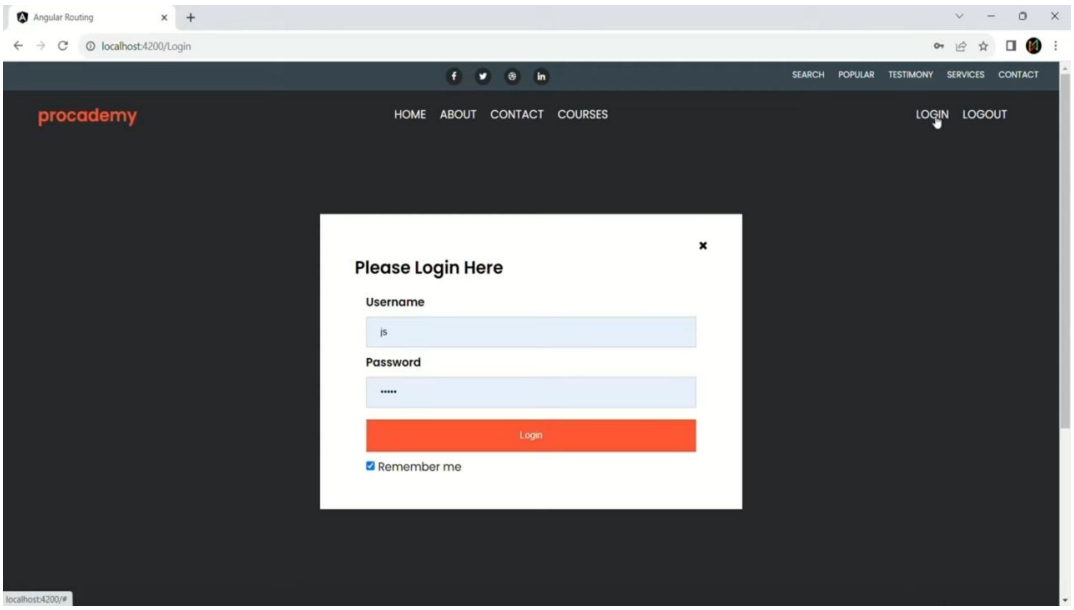


A new user model has been implemented with the ability to create a username and password.

3:06:08

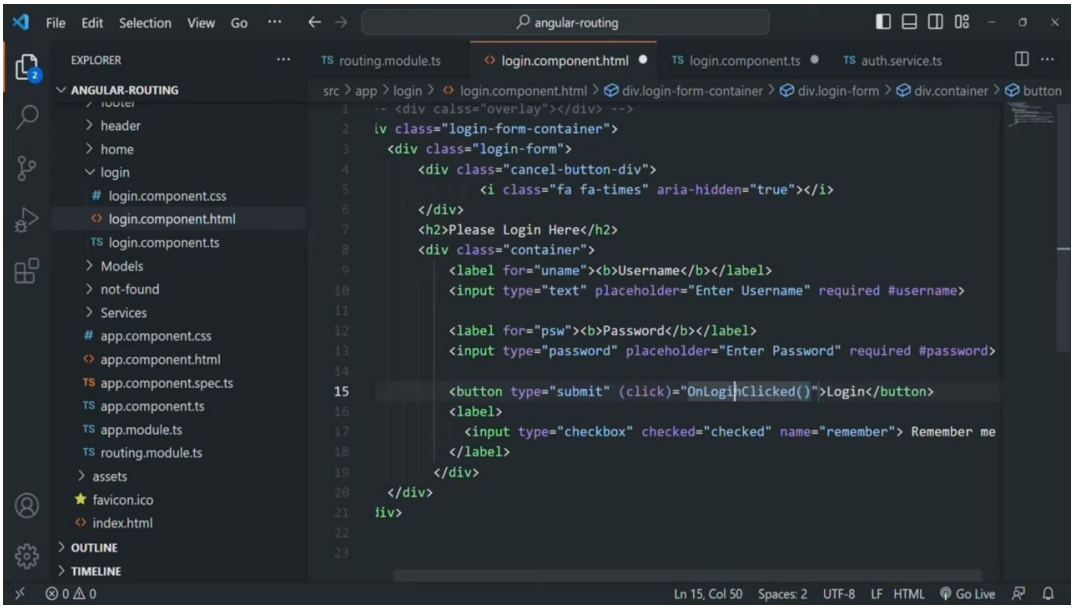


3:06:11



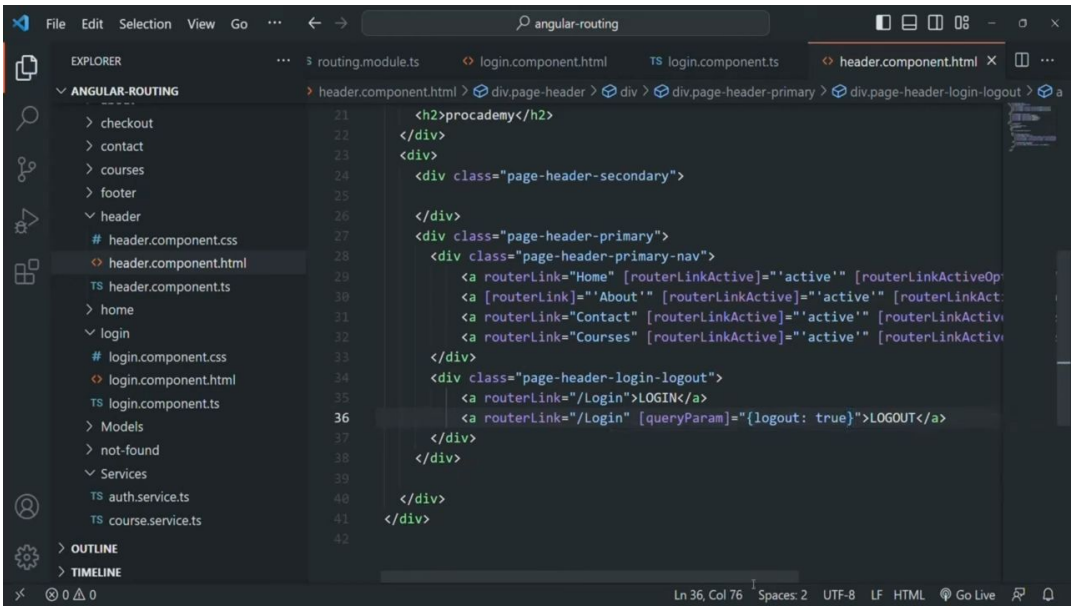
Developing a login page for a website involves integrating the login component with the header component.

3:14:41



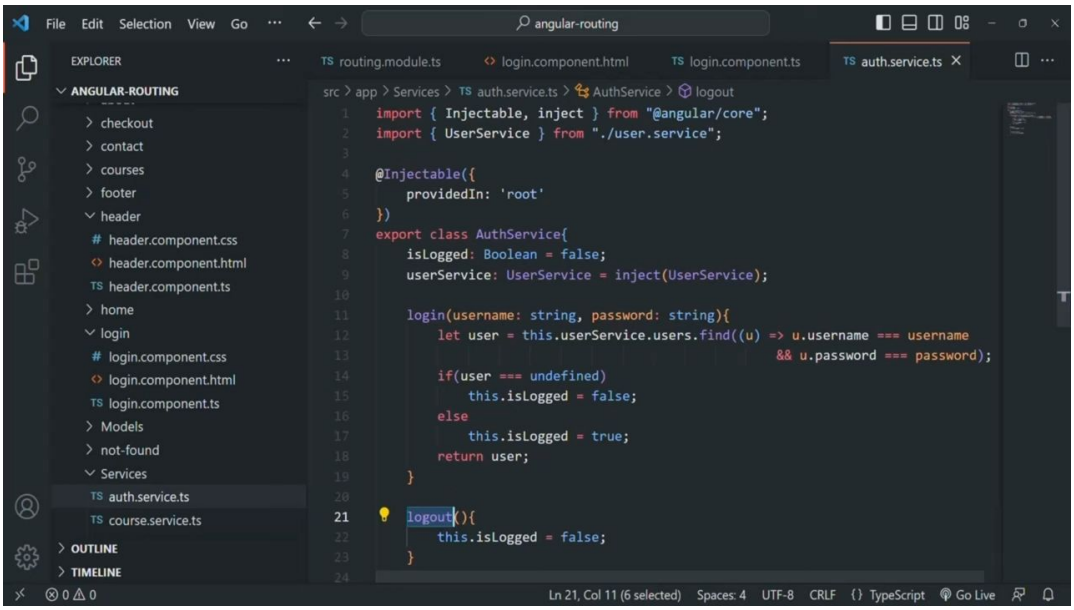
A new method for handling login button clicks has been added to the login component.

3:18:27



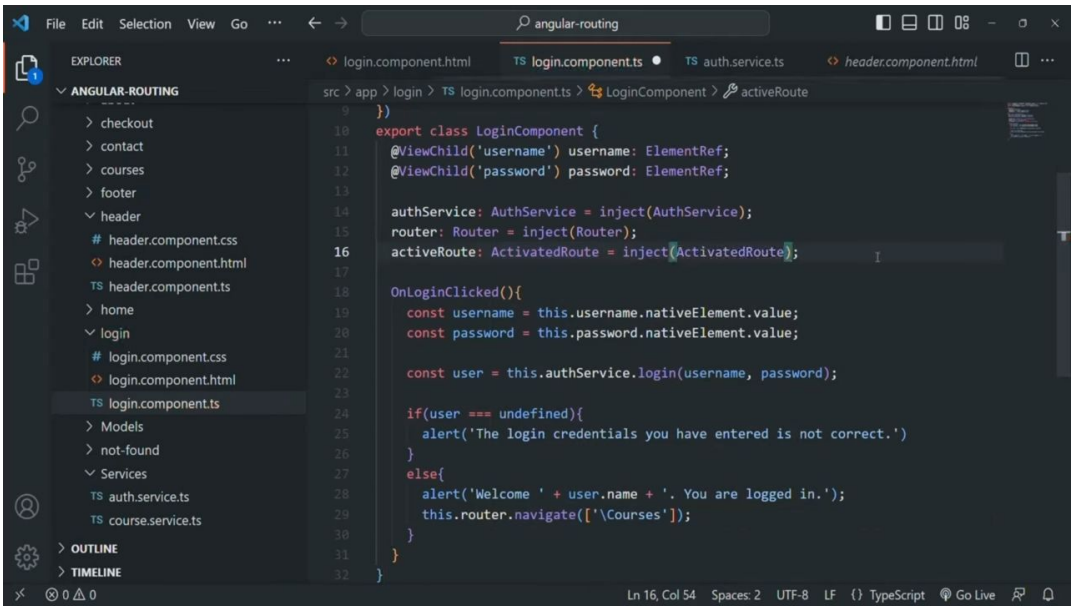
Developers are working to assign an object and specify a log out query parameter for a web page.

3:26:18



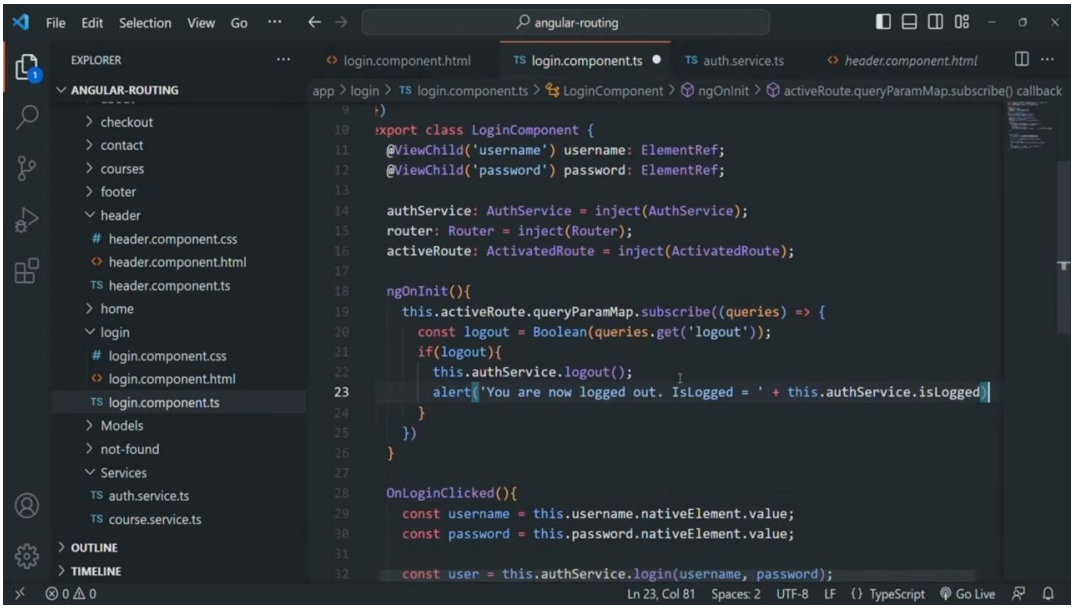
The logout method of the auth service will be called in the login component to read the query parameter and perform the necessary logic.

3:27:34



Angular developers are implementing the NG on init lifecycle hook to inject an instance of the activated route class inside the login component.

3:28:37

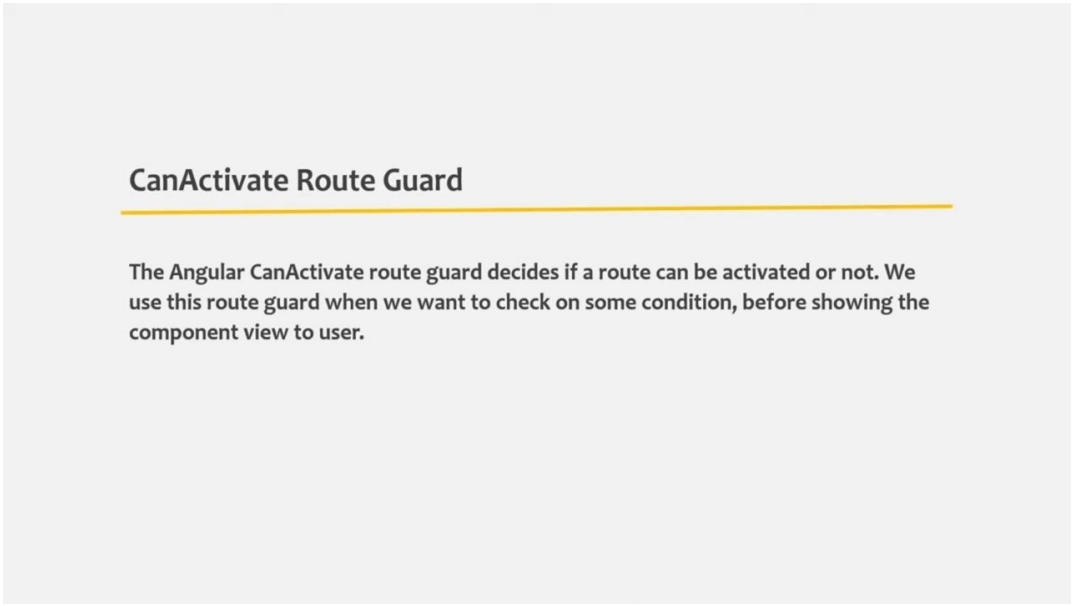


The authentication service's log property is being tested to ensure it is set to false when the user logs out.

3:31:03

In this article, we will explore how to implement route guards in Angular and understand the purpose of the can activate route guard.

3:32:05



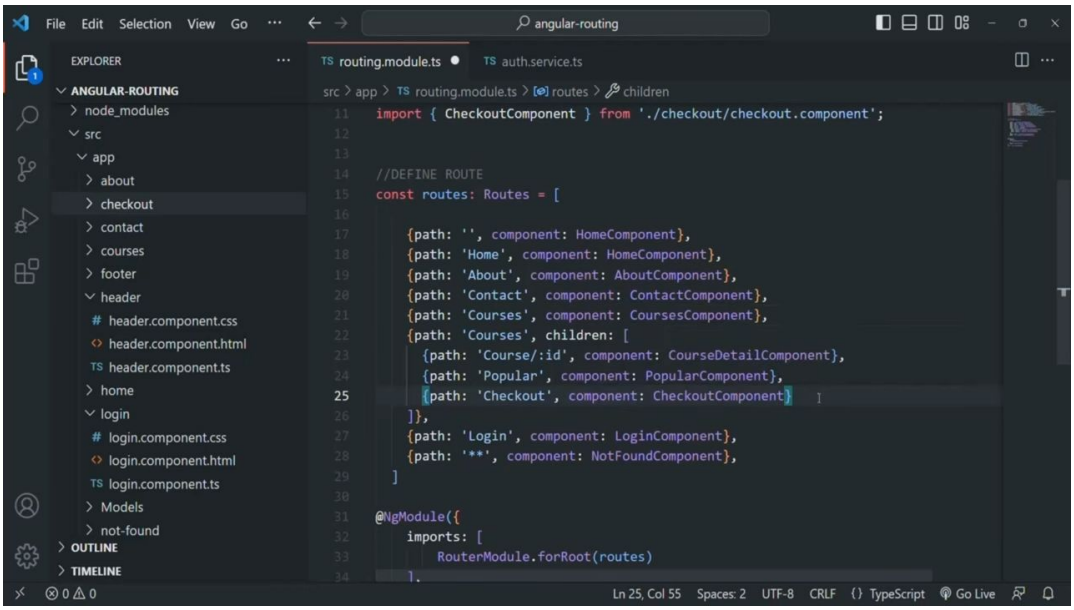
Learn how to implement route cards in Angular and understand the purpose of each route guard, starting with the can activate route guard.

3:32:27

CanActivate Route Guard Use Case

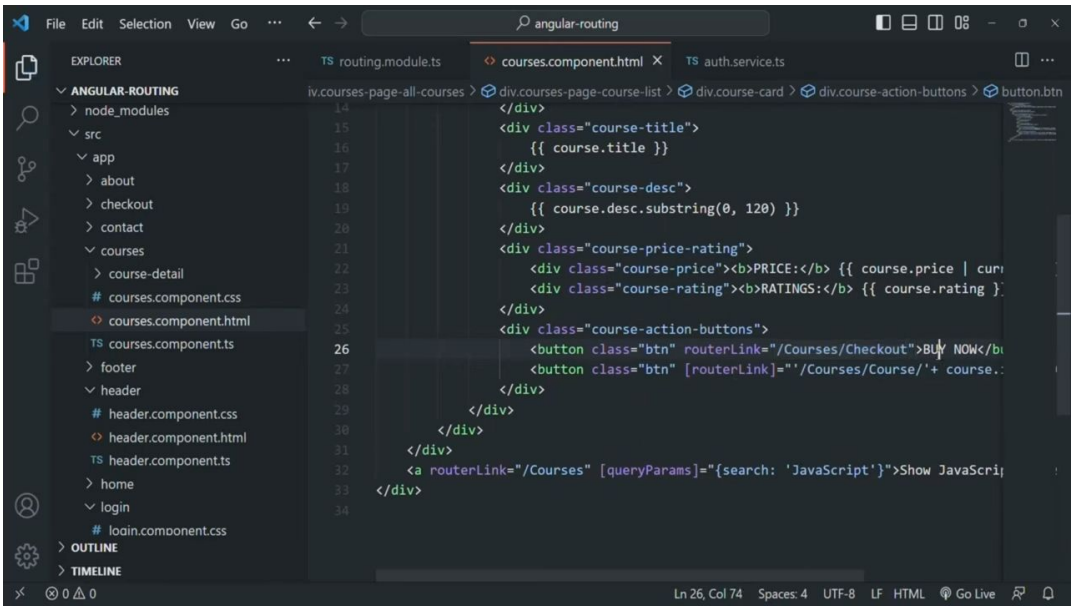
The common use case where we can use a CanActivate route guard is when we want to protect a route from an unauthorized user. For example, let's say there are some routes which can only be accessed by a logged in user, we can use CanActivate guard.

3:32:38



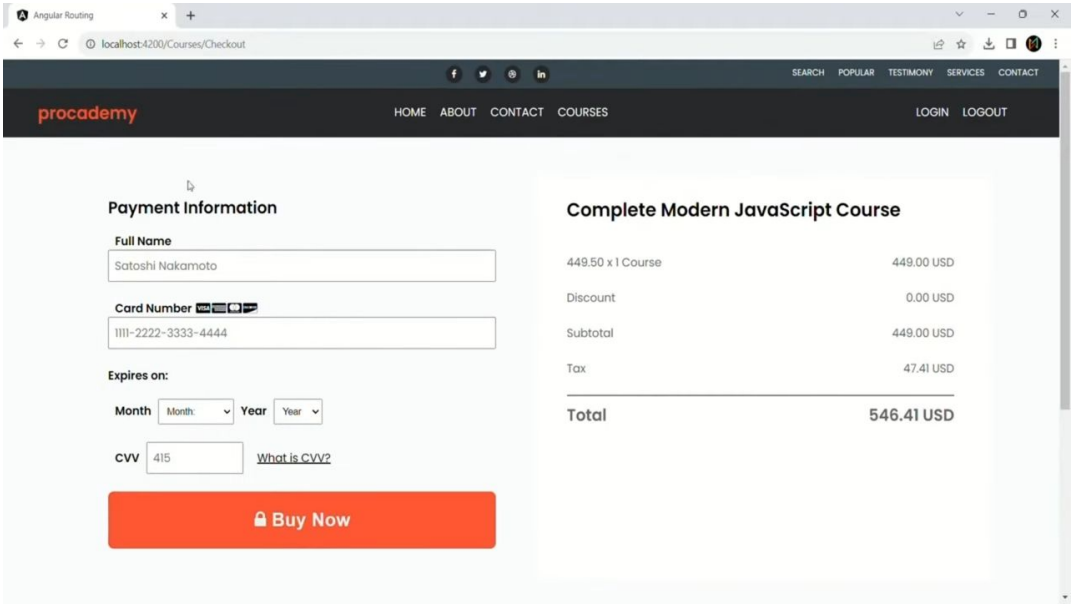
New route and component properties have been specified for the checkout process in the courses component.

3:35:15



New feature added to website allows for dynamic redirection to checkout page based on selected course.

3:35:48



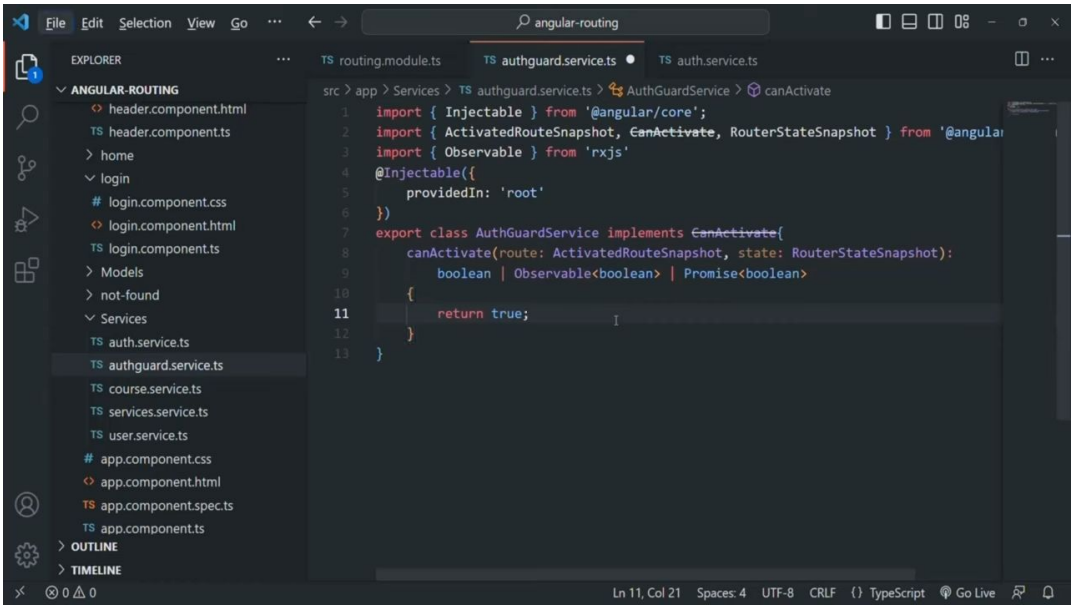
3:35:59

CanActivate Guard: Angular 14 & Lower versions

1. Create a service which inherits from CanActivate interface.
2. Implement CanActivate method provided by CanActivate interface.
3. Return a boolean value from the CanActivate method based on if the route can be activated or not.
4. Assign the service to canActivate property of route object.

New Angular 15 introduces a new way to implement route guards, offering flexibility for developers.

3:37:21



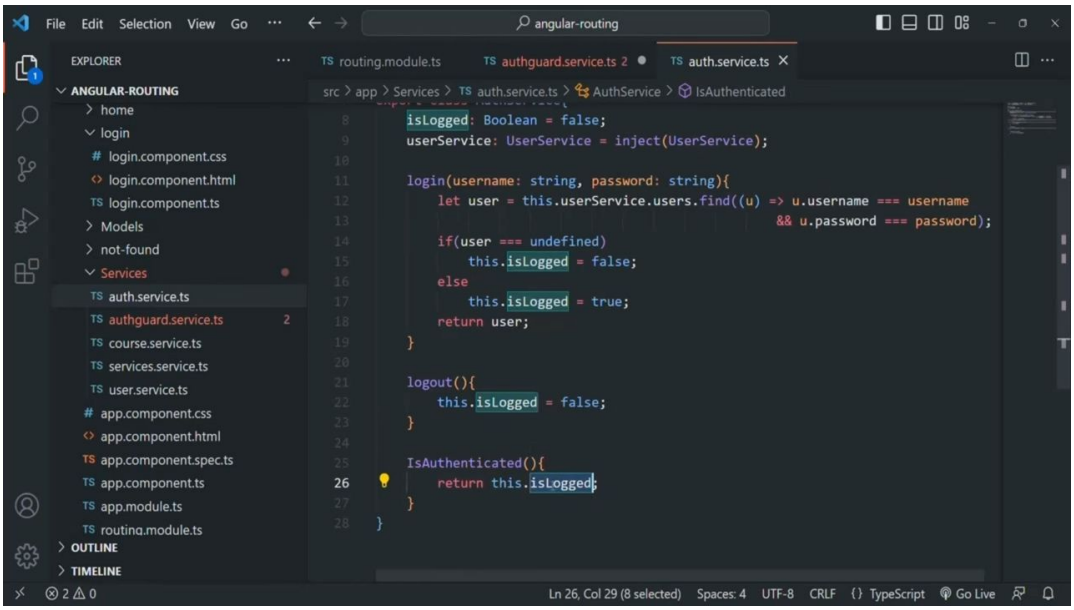
New service implementation successfully resolves connectivity issues and creates a route activation card.

3:42:51

CanActivate Guard: Angular 14 & Lower versions

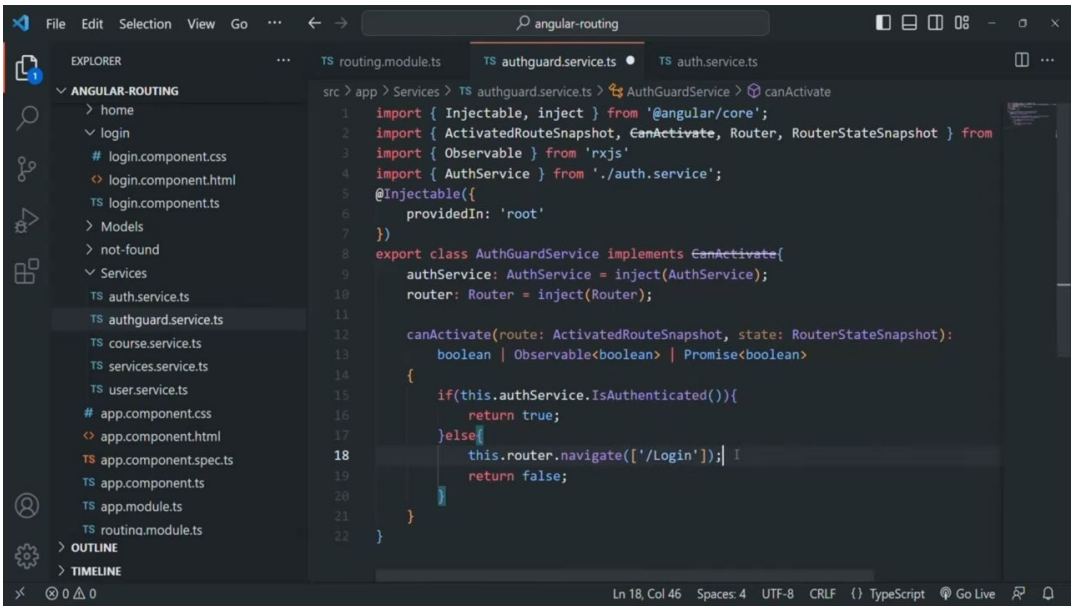
1. Create a service which inherits from CanActivate interface.
2. Implement CanActivate method provided by CanActivate interface.
3. Return a boolean value from the CanActivate method based on if the route can be activated or not.
4. Assign the service to canActivate property of route object.

3:42:55



Service dot is authenticated method returns a Boolean value based on the logged status.

3:47:56

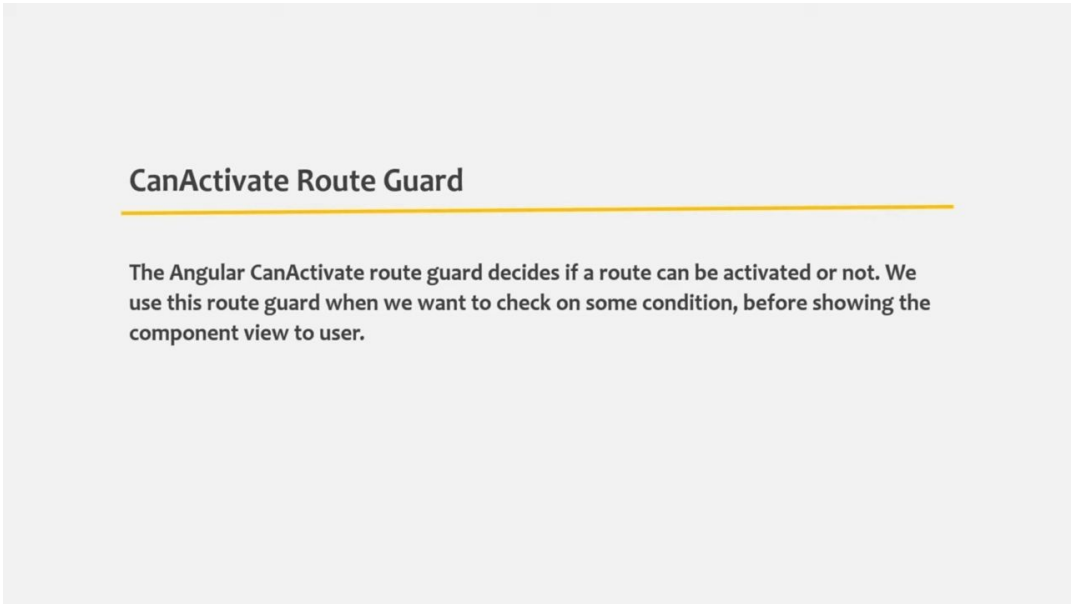


Developers are implementing a new navigation feature to direct users to the login page on a web application.

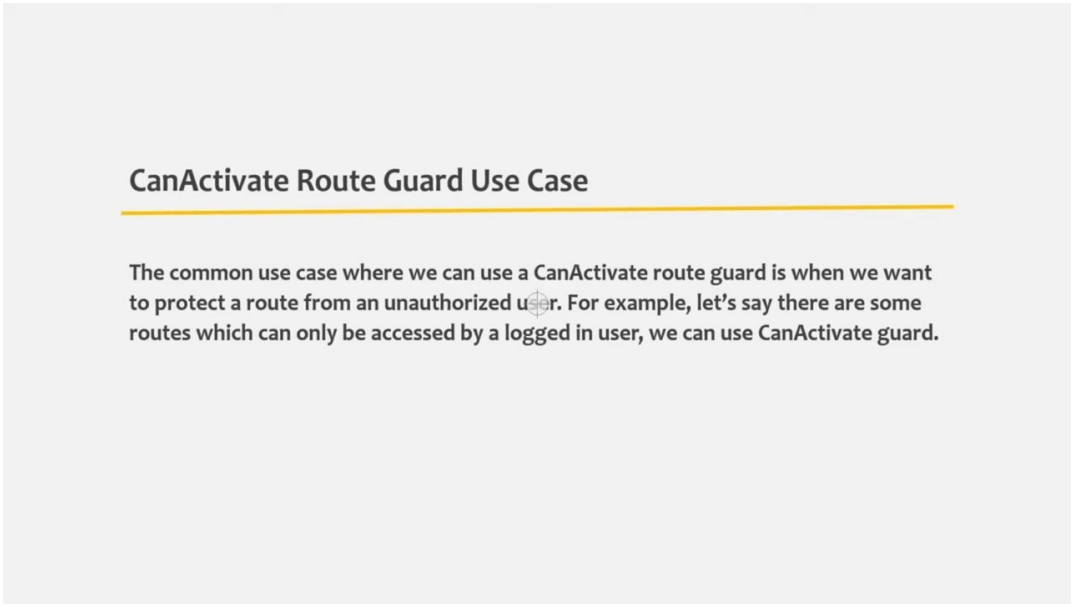
3:49:23

Learn how to implement can activate route guard in Angular 15 and higher versions in the next lecture.

3:51:24

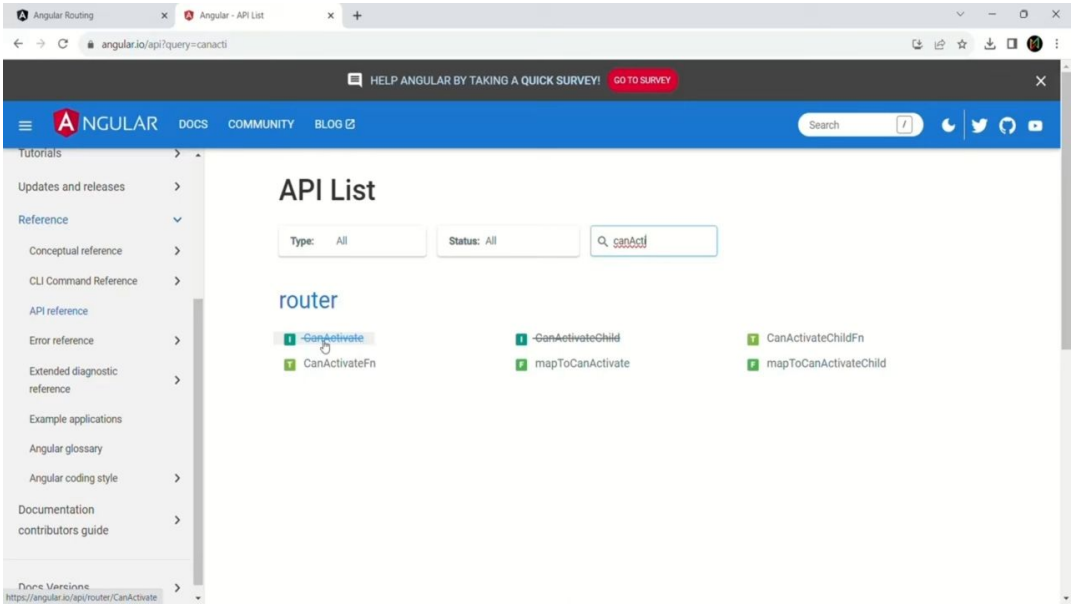


3:51:30



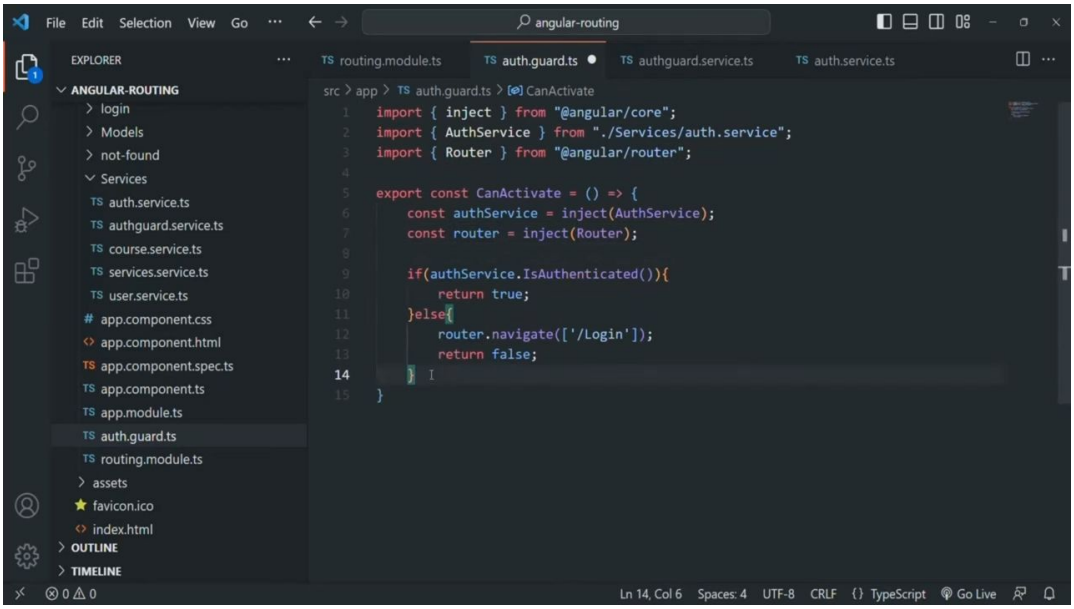
New Implementation Method for Connectivate Cloud in Angular 15 and Higher Versions.

3:51:57



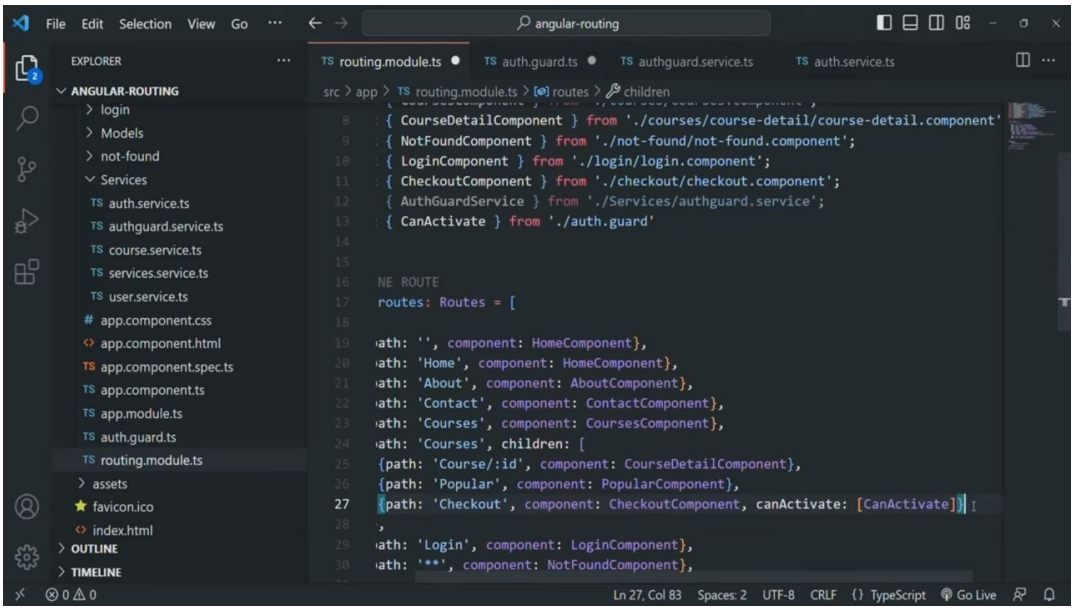
Angular has deprecated the connectivate function in favor of the new can activate function in version 15 and higher.

3:53:05



Accessing router variables and importing functions for routing modules.

3:59:00



New update allows for seamless activation of connectivity function in application, ensuring continued smooth operation.

3:59:50

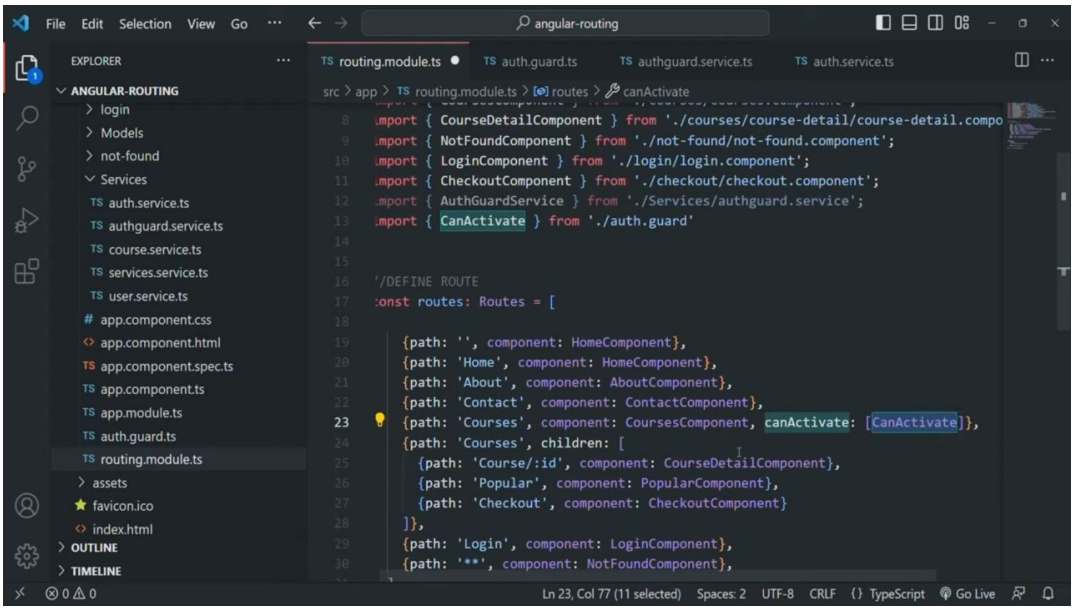
Learn about the Connectivate Child Route Guard and its implementation in Angular.

4:02:13

CanActivateChild Route Guard

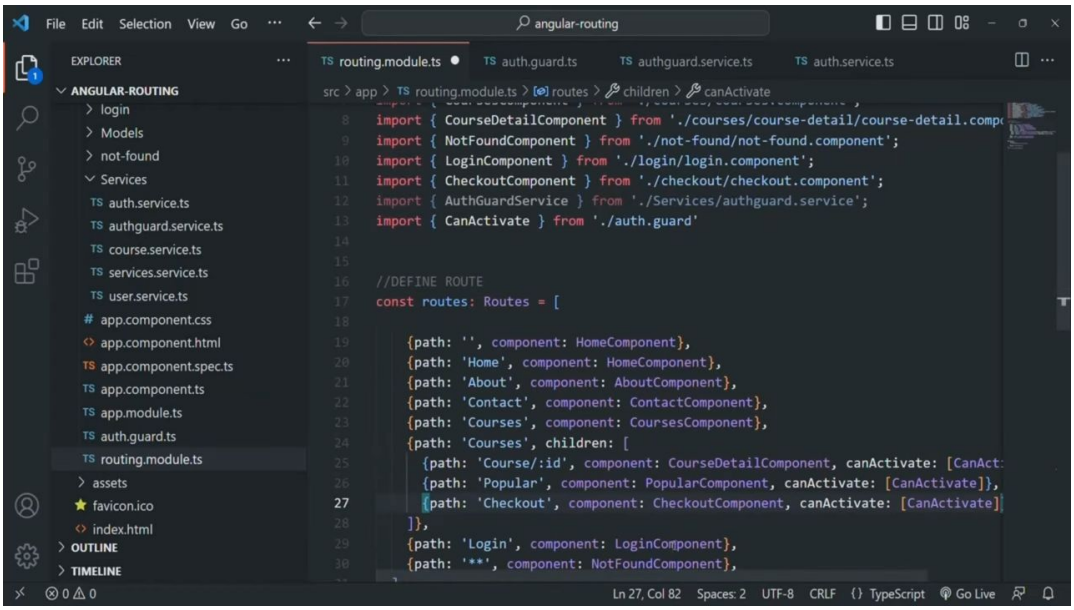
The Angular CanActivateChild route guard runs before we navigate to a child route guard. This guard is very similar to CanActivate route guard and it protects a child route from unauthorized access.

4:02:22



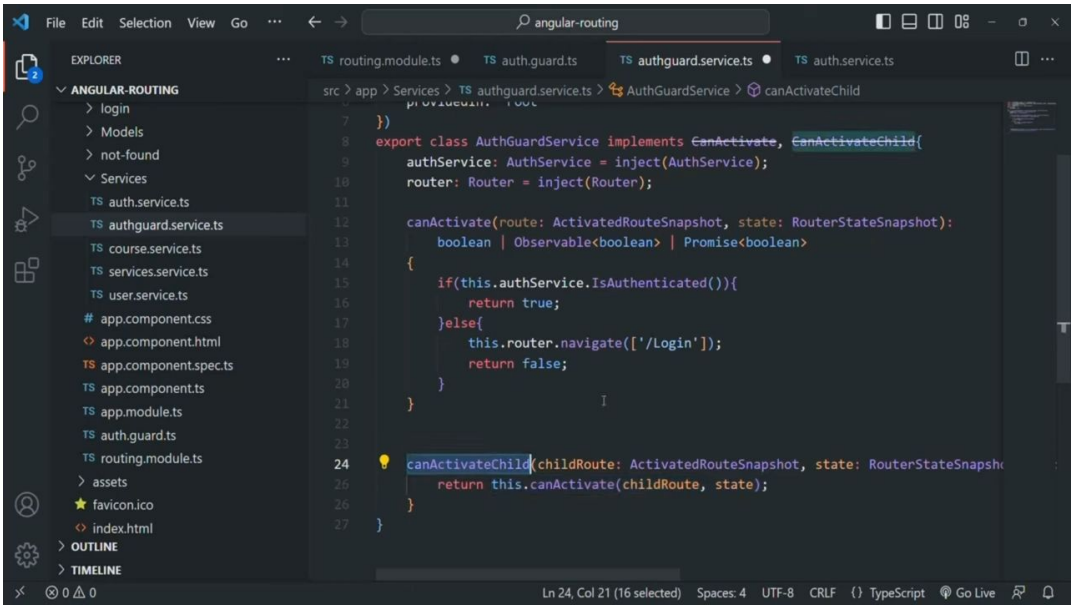
Connectivate can be used to protect both parent and child routes on a course route.

4:03:18



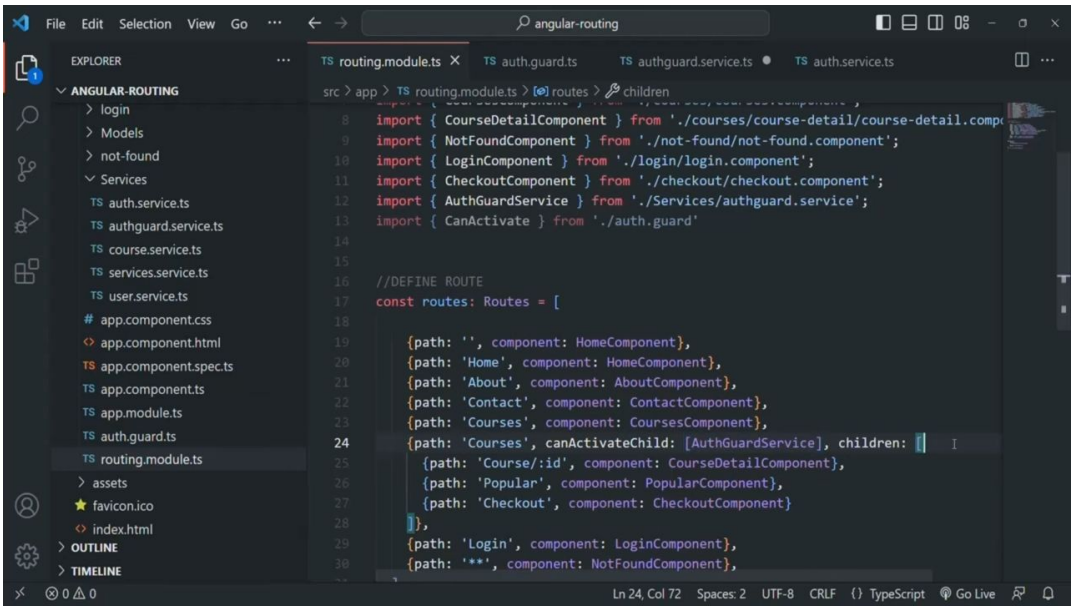
New feature allows for more granular access control on child routes within an application.

4:04:18



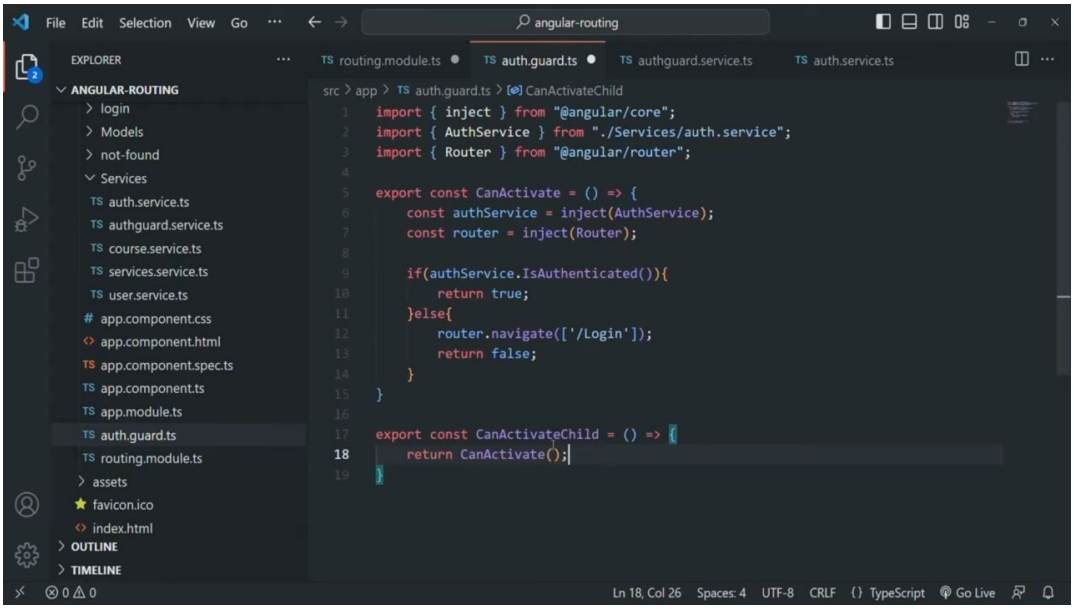
Auth Guard Service now implements the Connectivity Child interface, allowing for seamless assignment to the can activate child property.

4:10:17



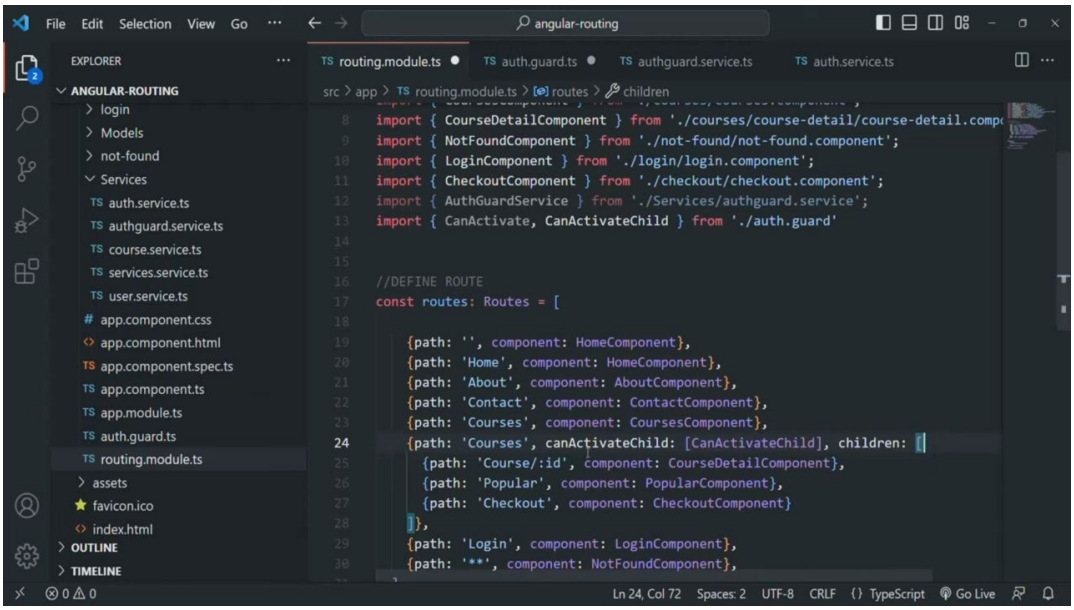
New security measures implemented to restrict access to certain web pages.

4:10:58



The connectivity function returns a Boolean value and is imported into the routing module.

4:15:12

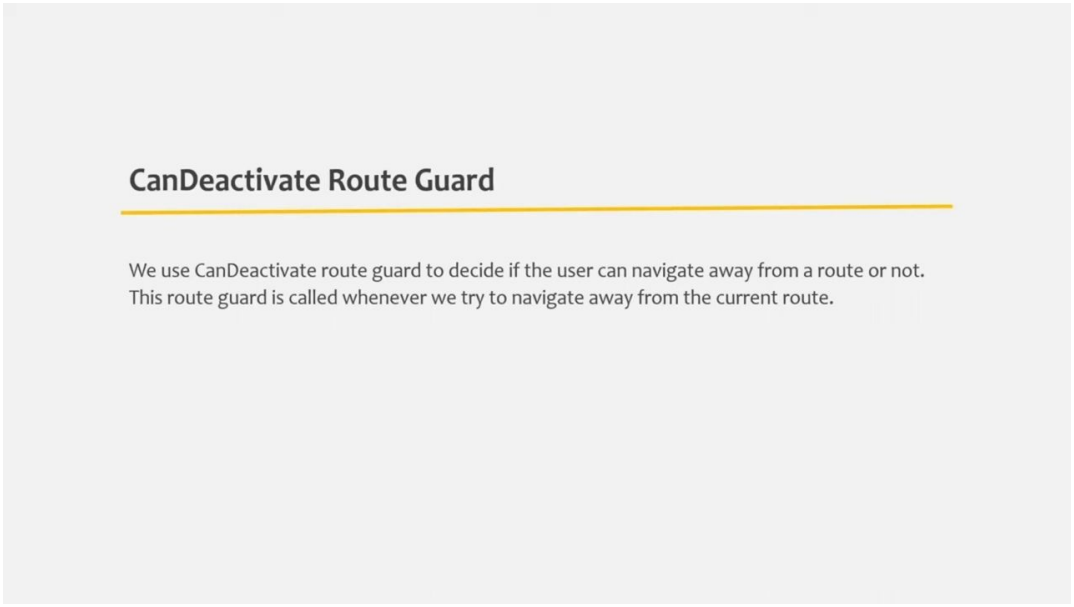


New feature in Angular allows for protection of child routes using the connectivate function.

4:15:34

Learn about the can deactivate route guard in Angular and how it can be used to protect routes from navigation.

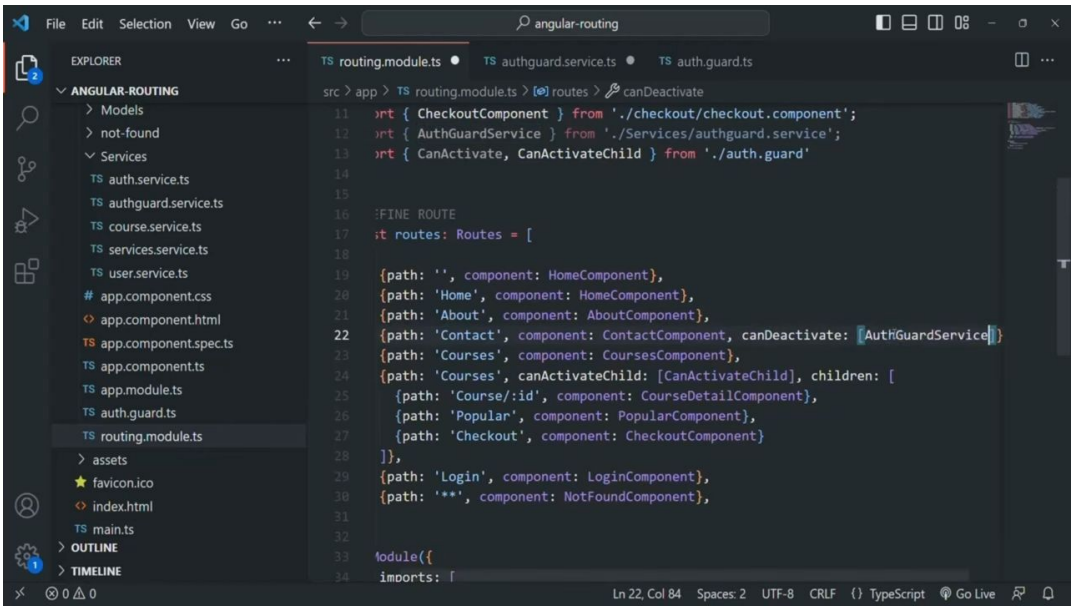
4:17:12



4:17:26

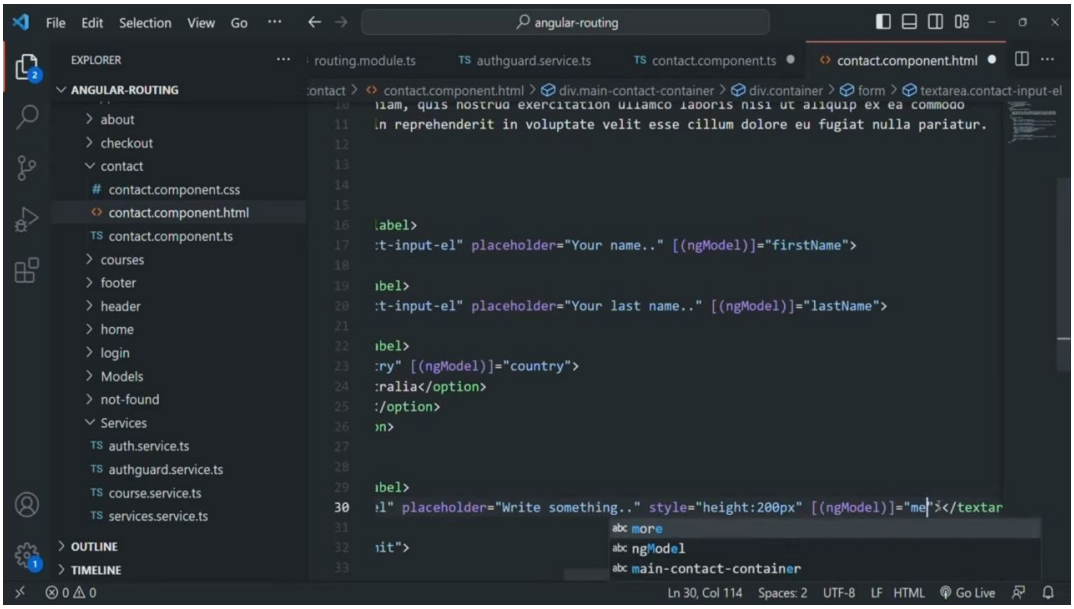
A can deactivate route guard is essential for preventing users from navigating away from a form before submitting it.

4:17:34



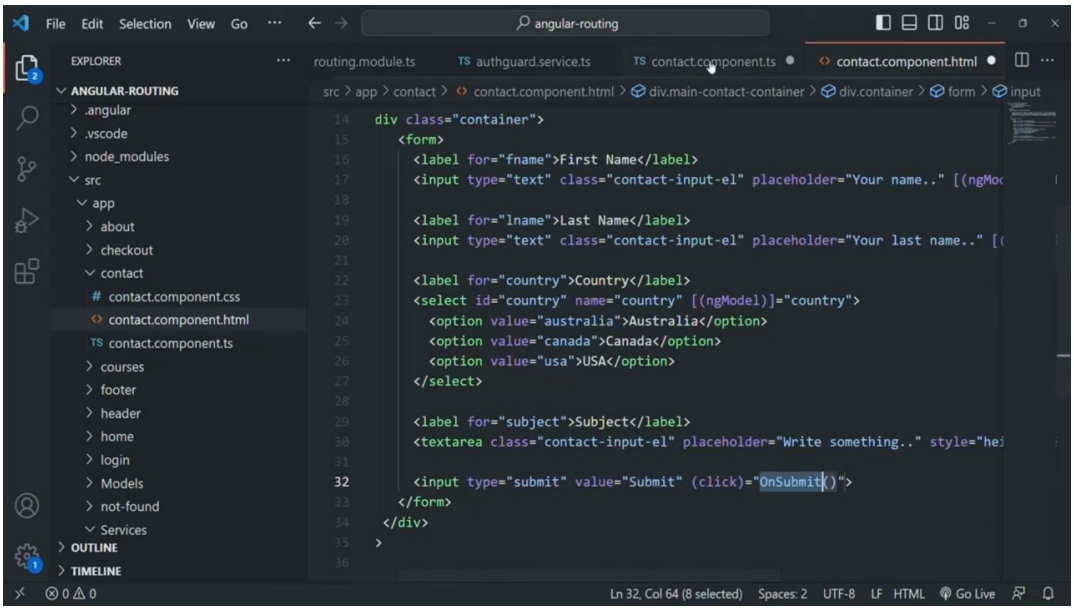
Property assignment and implementation of route guards for the contact page.

4:23:33



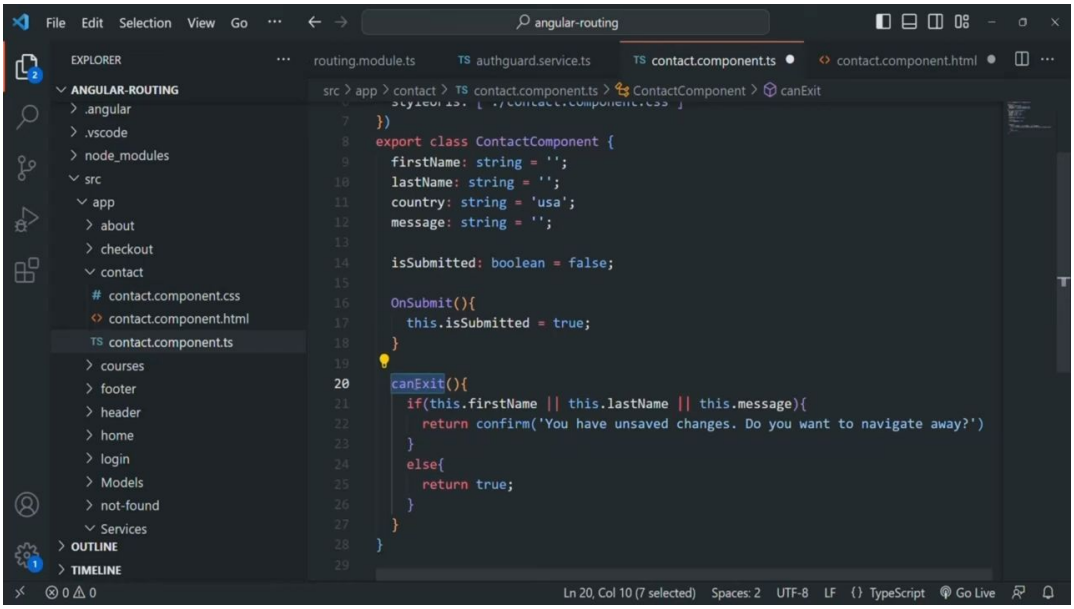
Angular developers are now able to bind text areas to message properties using the NG model and forms module.

4:27:57



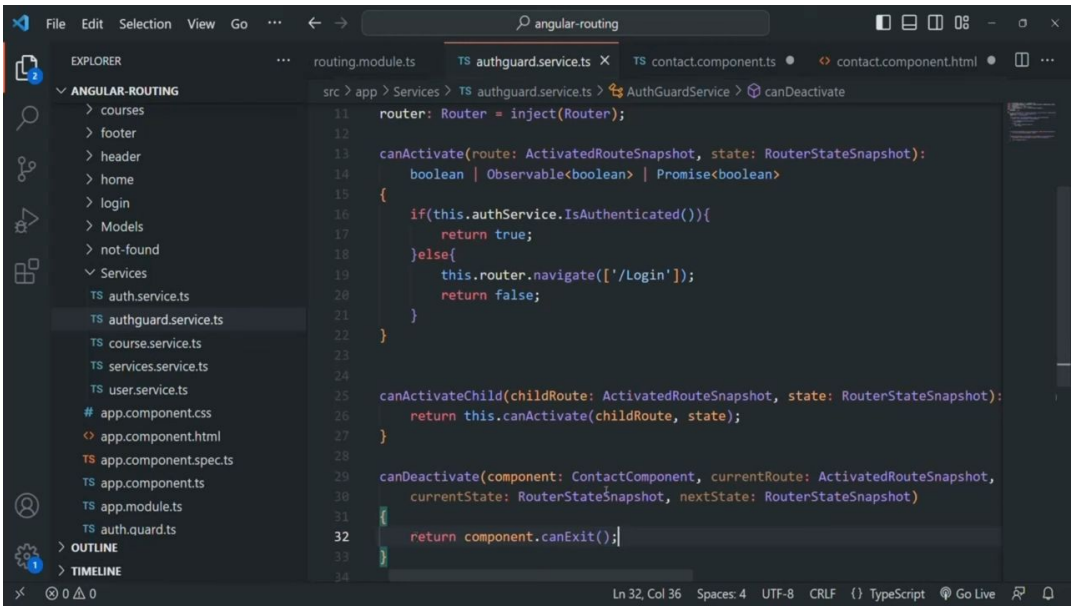
A new method for handling form submission and navigation has been introduced in the latest component class.

4:29:37



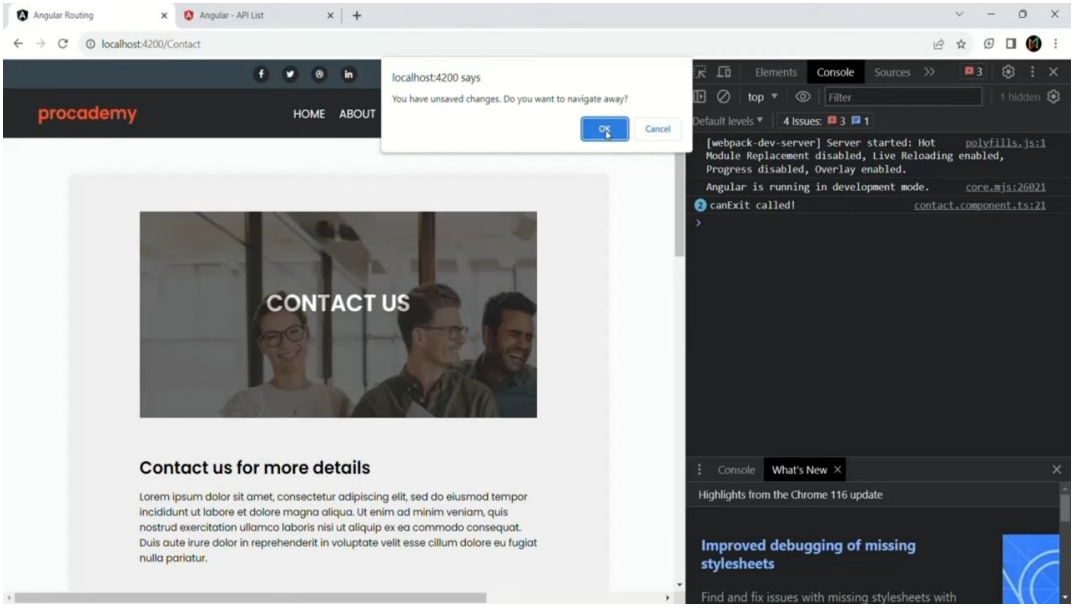
New method implemented in contact component for activating route card.

4:32:16



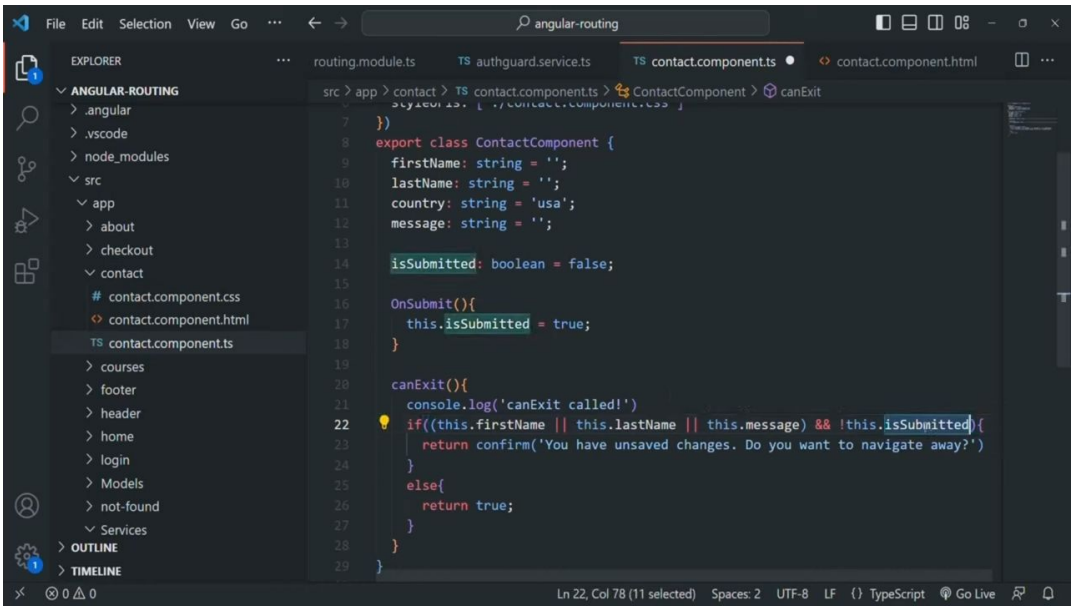
New implementation on the contact page allows users to navigate away without filling out the form.

4:32:46



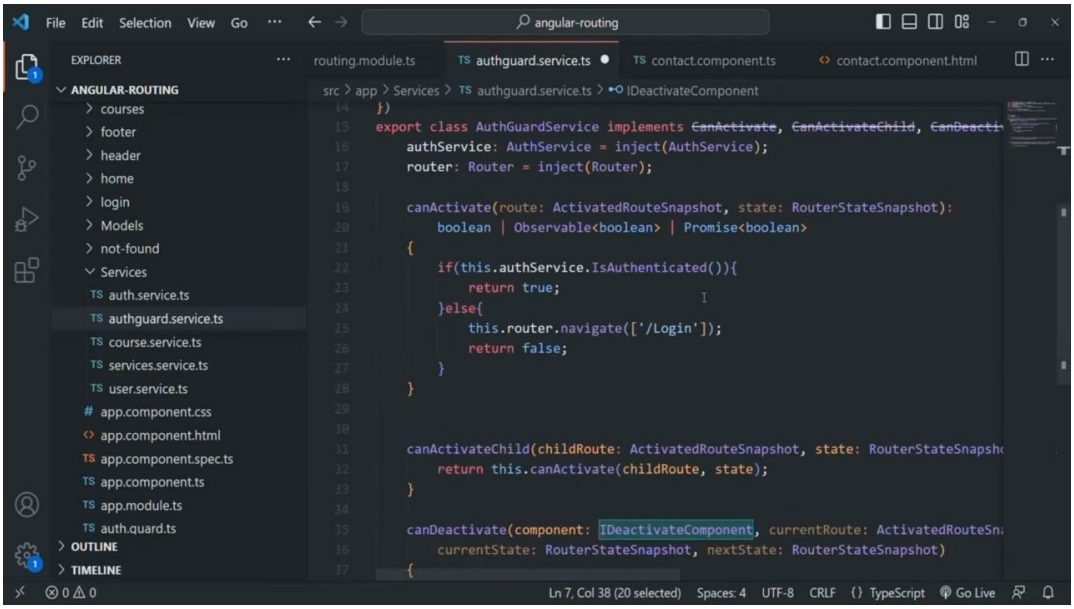
Navigating away from unsaved changes in a form can be easily managed using the deactivate route card.

4:34:19

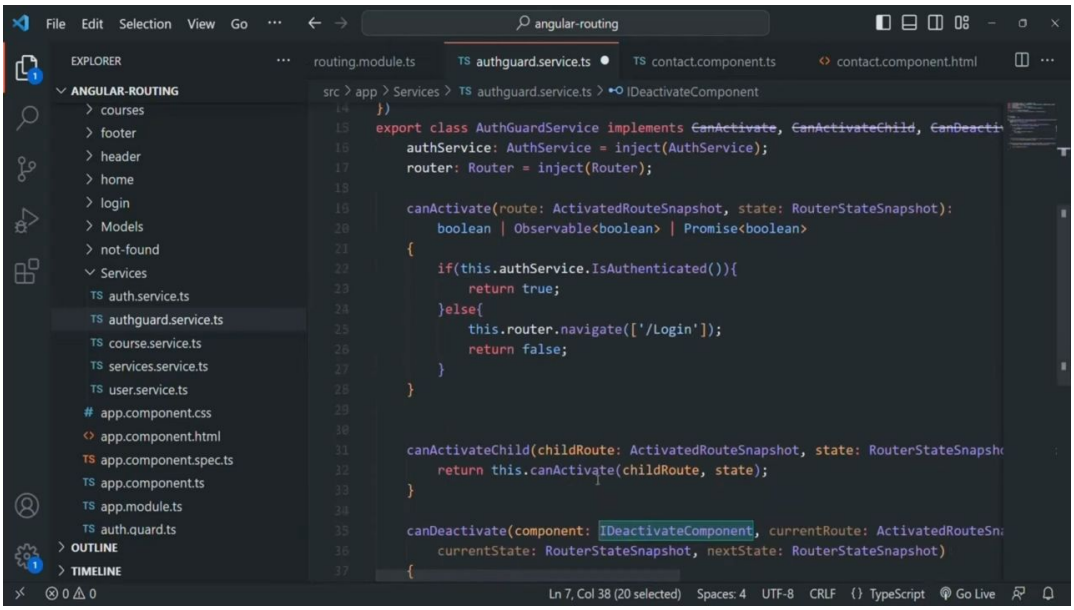


New feature in author guard service.ts file prevents users from navigating away without saving changes.

4:35:51

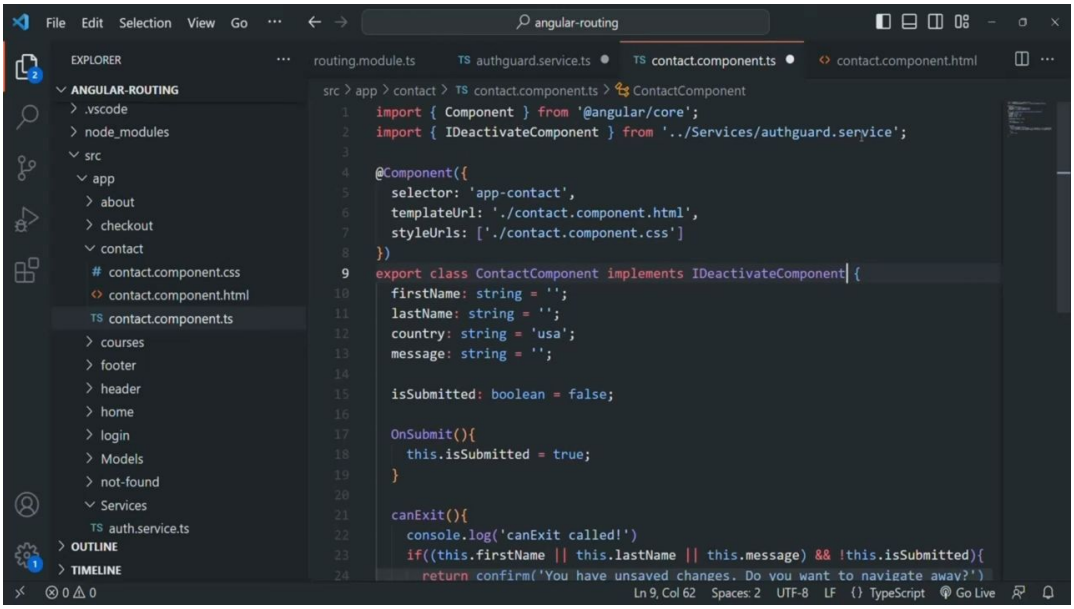


4:38:39



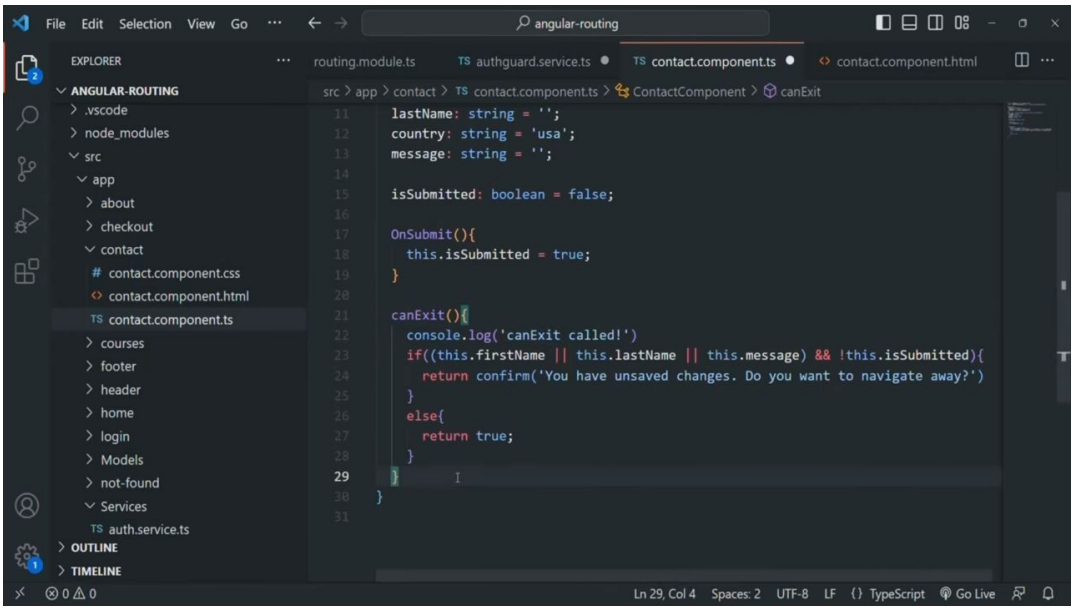
Developing a contact component that inherits the can deactivate route functionality from the IDeactivateComponent interface is essential for ensuring smooth navigation within the application.

4:38:39



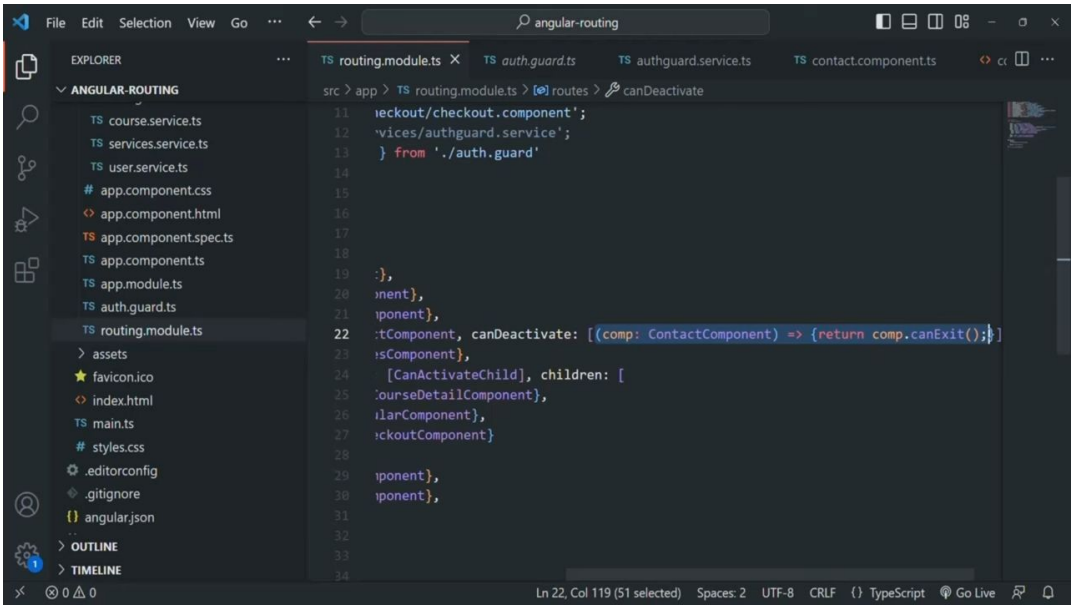
Component inheritance is achieved through the use of the "implements" keyword and implementing from a specific interface.

4:38:56



Software developers are implementing a new feature to prompt user confirmation before navigating away from a page.

4:39:14



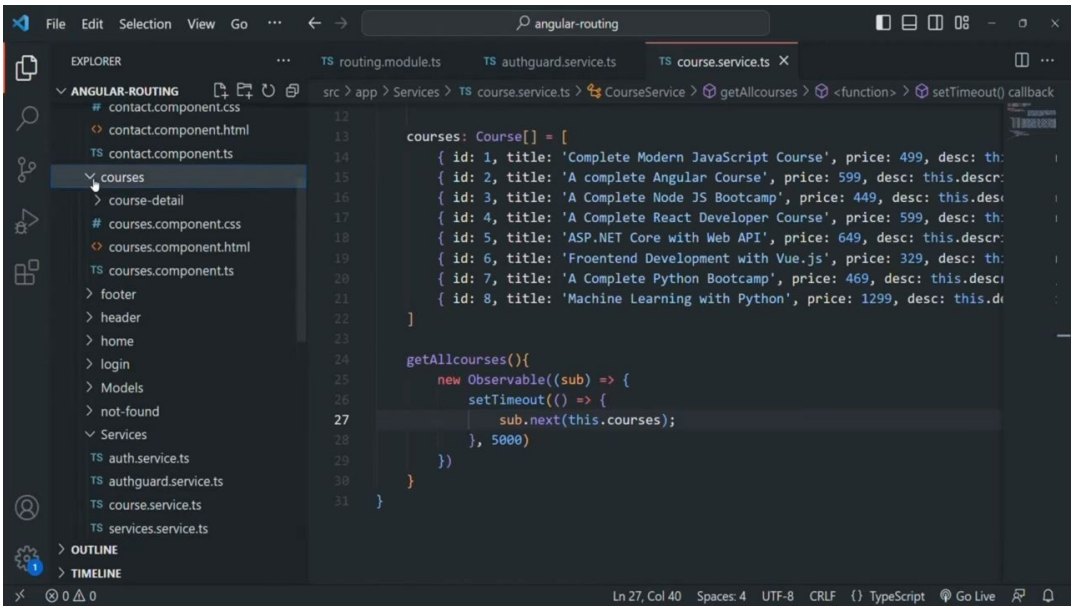
Deactivate route card is a feature used in Angular applications to implement the can deactivate function.

4:42:42

Learn about the resolved route guard in Angular and how it can be used to load data before navigating to a route.

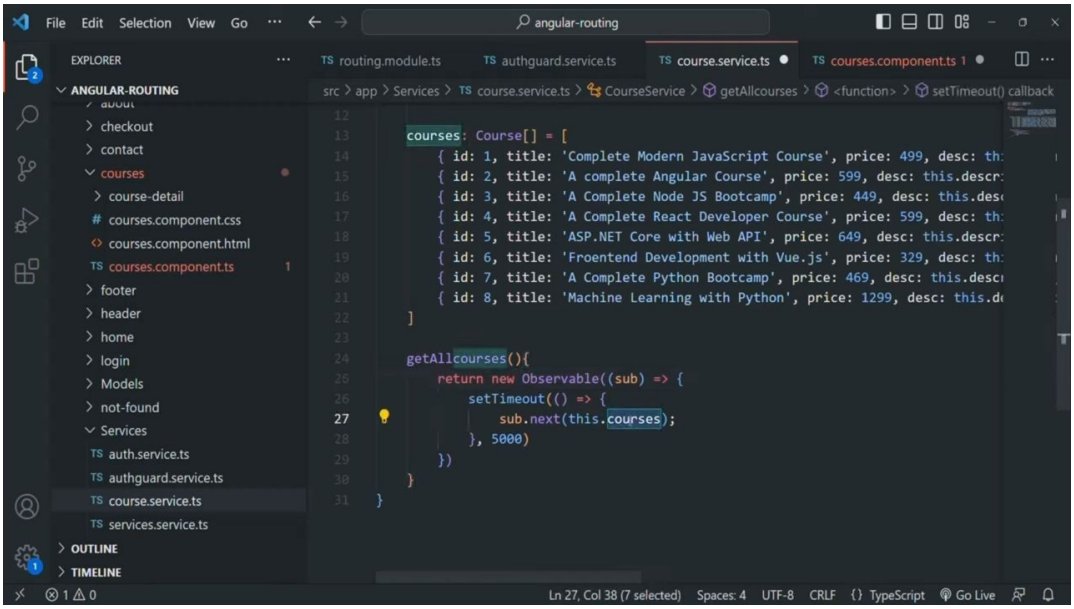
4:43:05

4:43:10



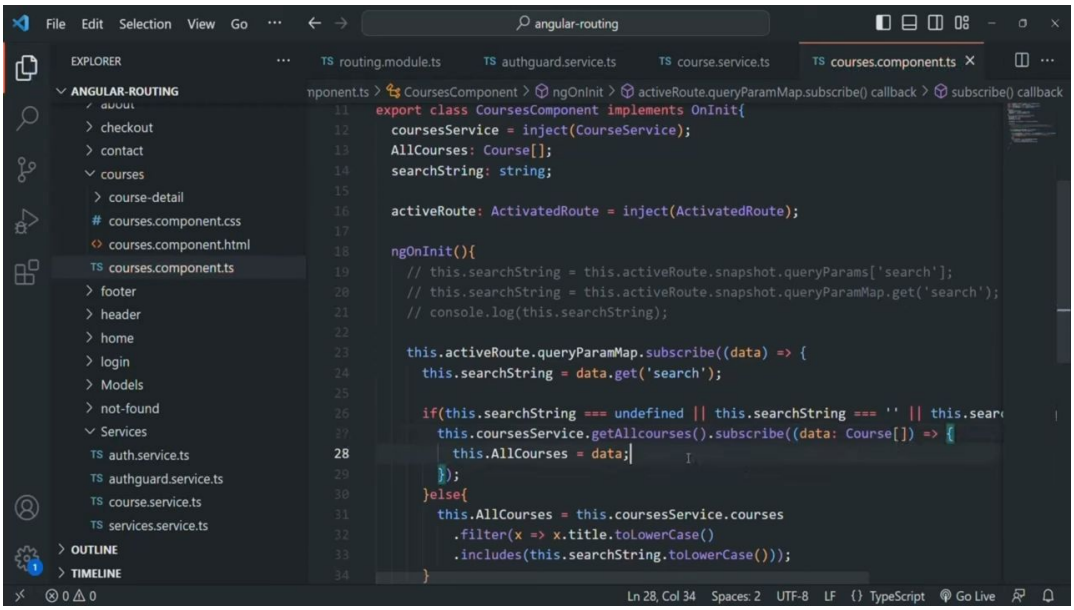
The courses component uses the "all courses" property to display course information.

4:46:27



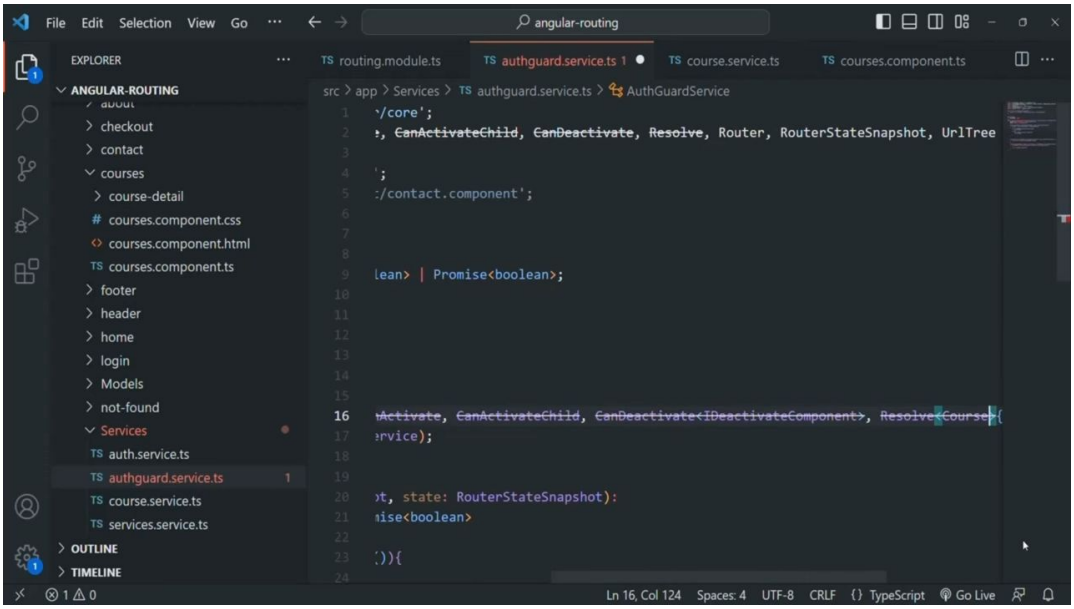
The observable is set to emit a list of courses, but the data parameter type needs to be specified in order to fix an error in the code.

4:47:43



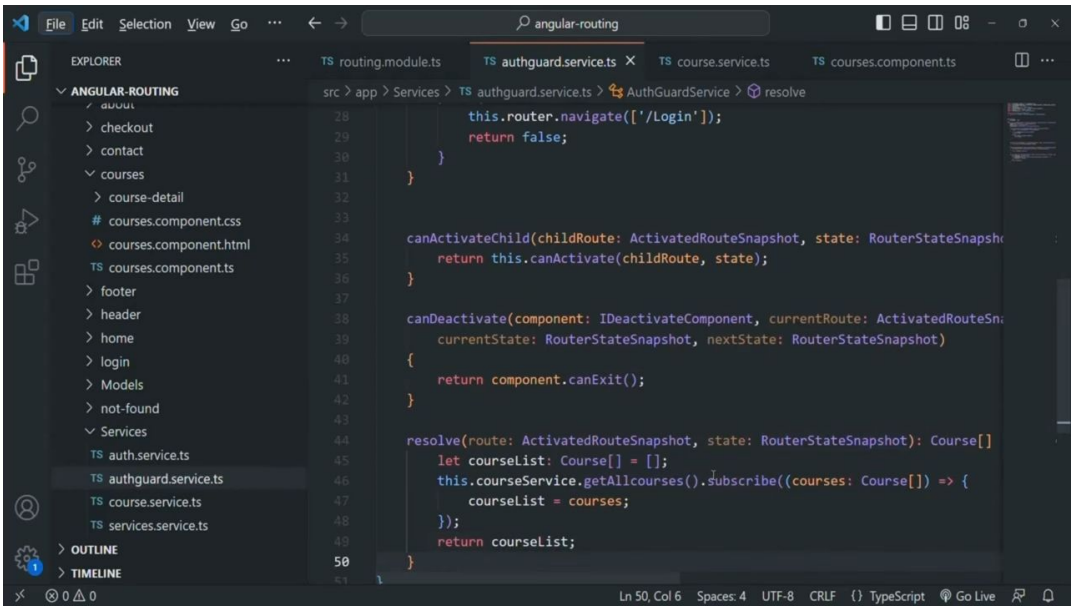
The course list on the web page will be available after five seconds of navigating to the courses link.

4:48:09



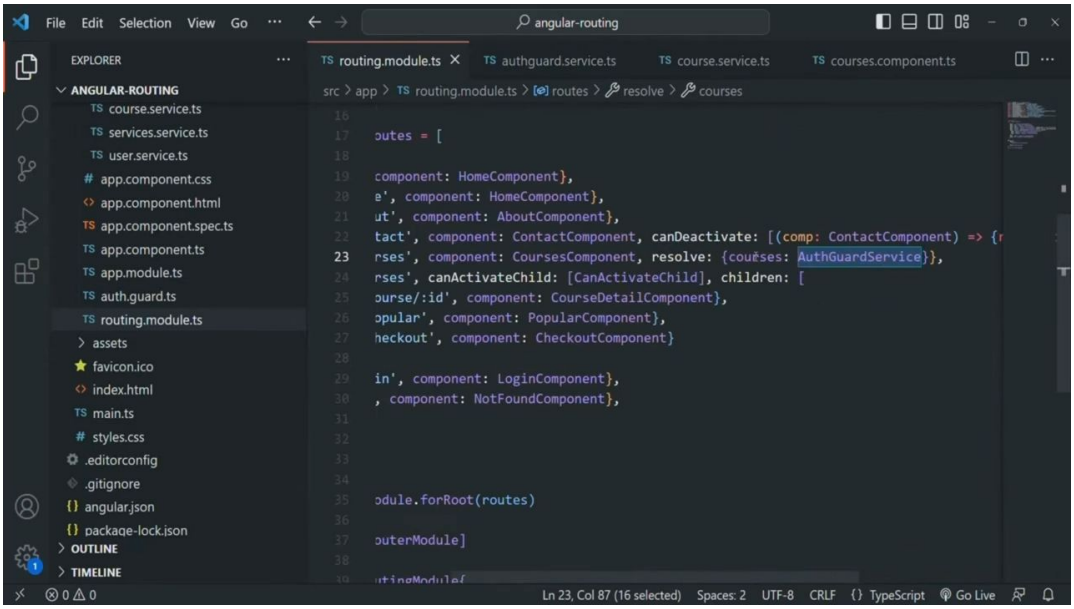
Service classes inheriting from resolved route guards must specify the type of data they will emit, such as an array of courses.

4:51:09



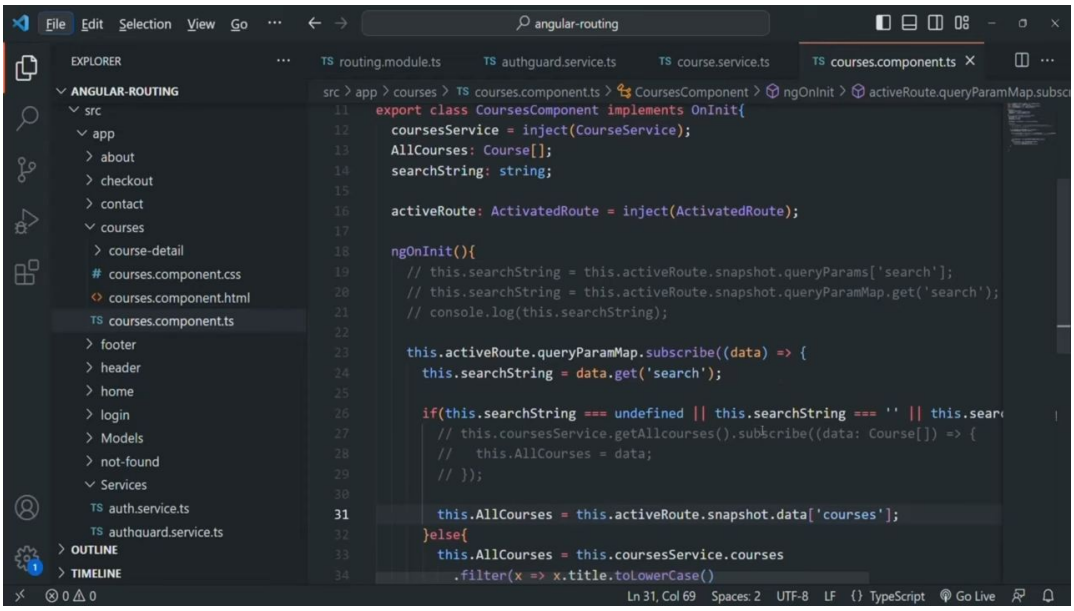
Resolve guards in Angular can be used to load data before navigating to a route by using the let keyword instead of const and returning the course list from the resolve method.

4:53:55



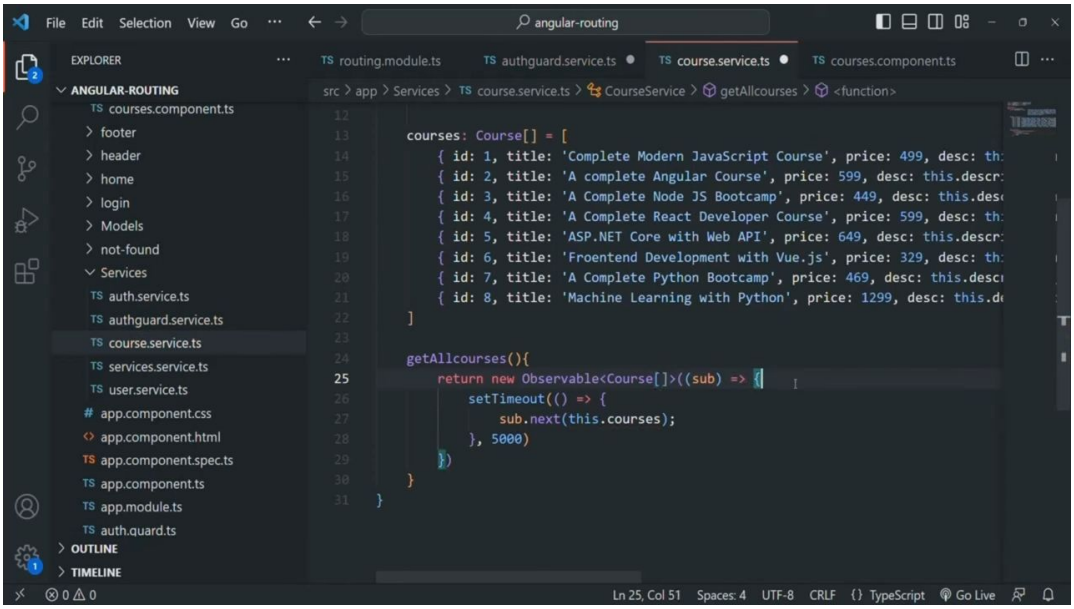
New Auth Card Service Implements Resolve Interface for Course Data Assignment.

4:55:37



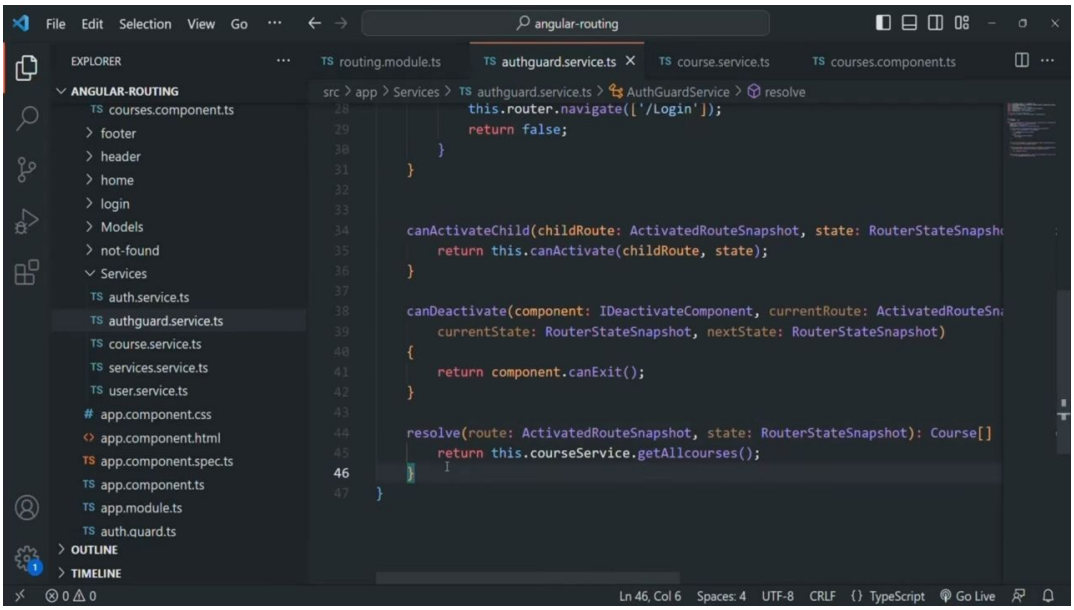
Website Redirect Issue Causes Course List to Disappear.

4:57:46

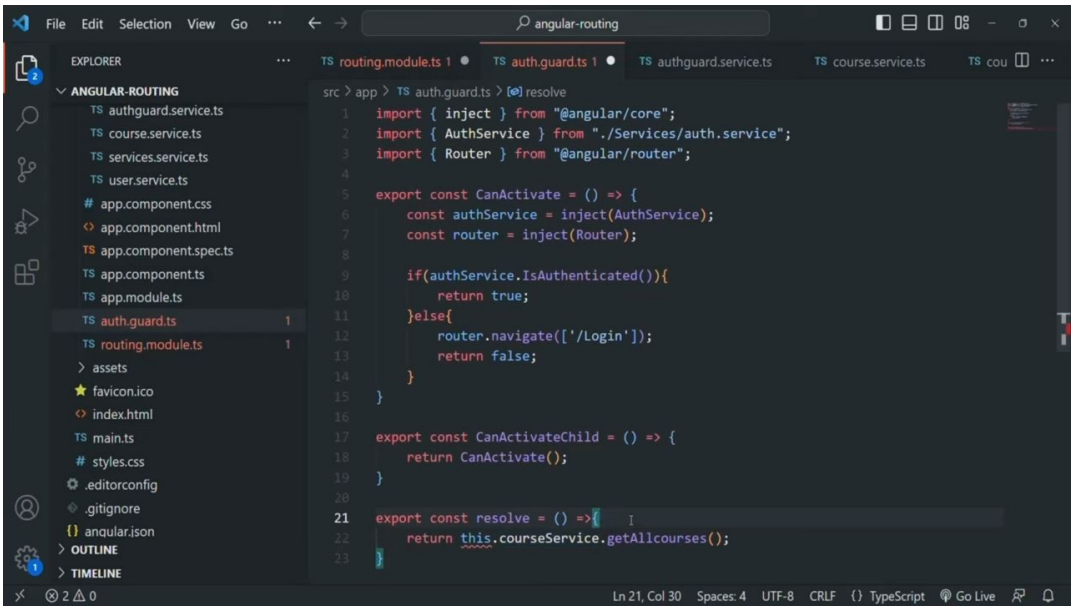


Developing observable data types for a web page.

5:00:04

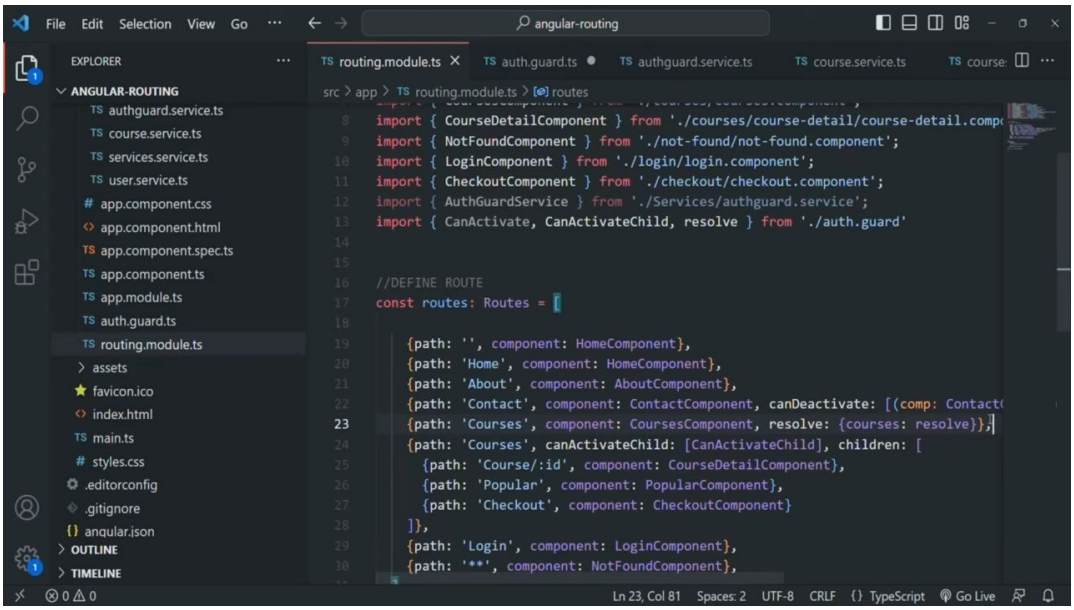


5:00:17



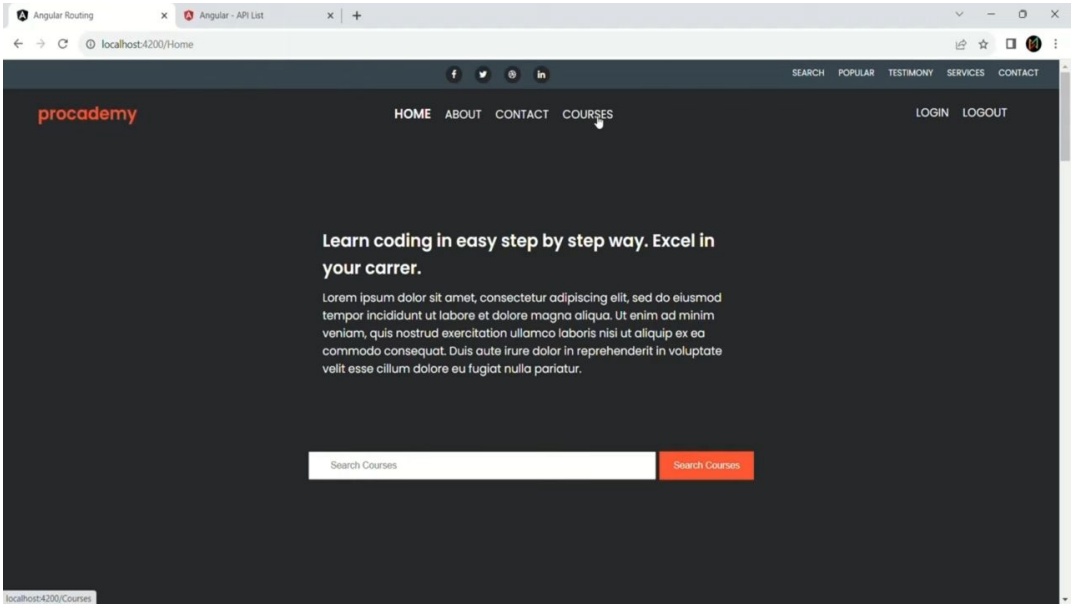
In the resolve method, we create a variable called course service and ask Angular to inject an instance of this service.

5:03:23



New functionality has been added to the courses page, allowing for a delay in displaying data.

5:04:16

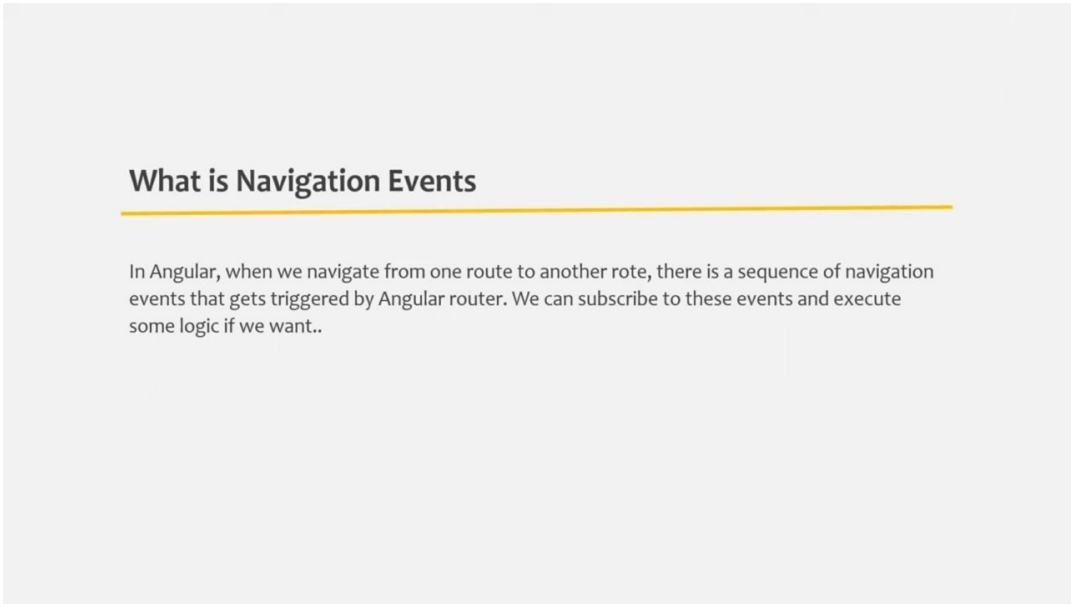


Resolve route card is a useful tool for ensuring that data is available before navigating to a specific page on a website.

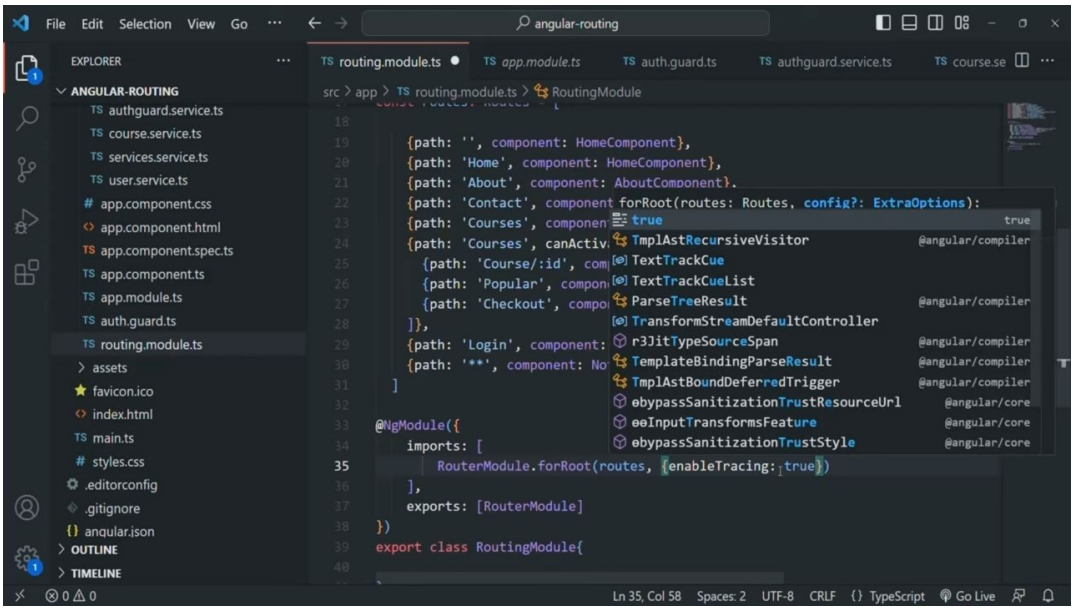
5:04:33

Learn how to implement a loading spinner during data loading and understand navigation events in Angular.

5:05:21

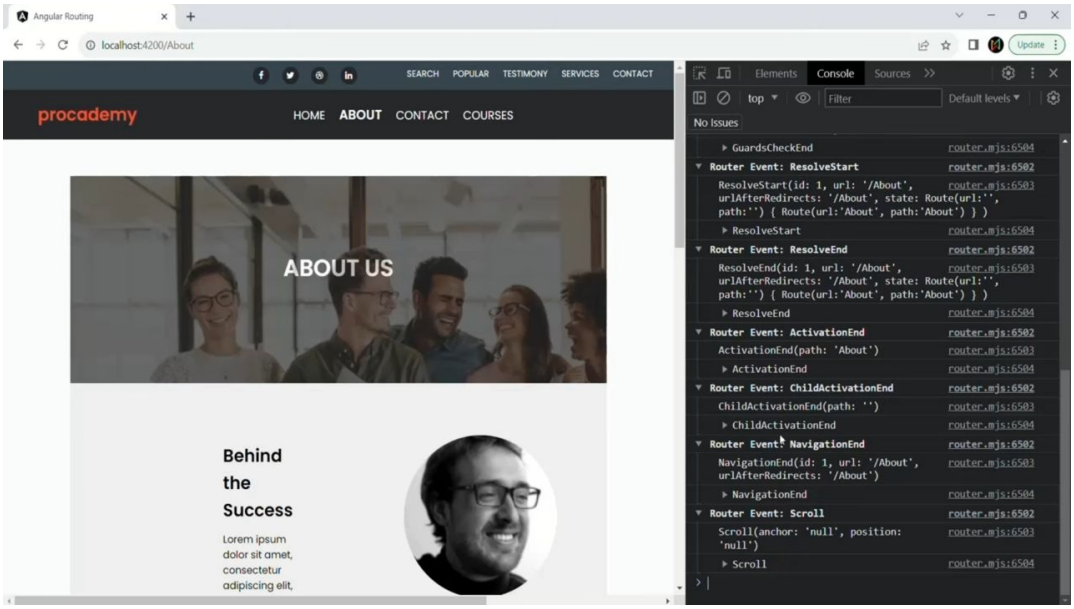


5:05:32



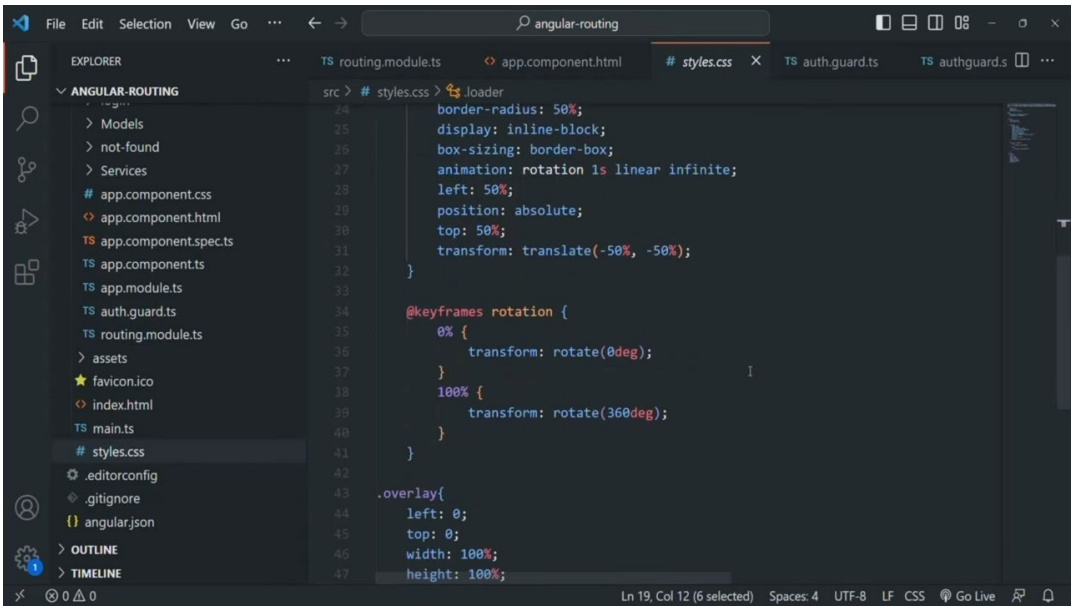
Developers can enable tracing to log navigation events in the browser's developer console by specifying an "enable tracing" property as true in an anonymous object.

5:06:46



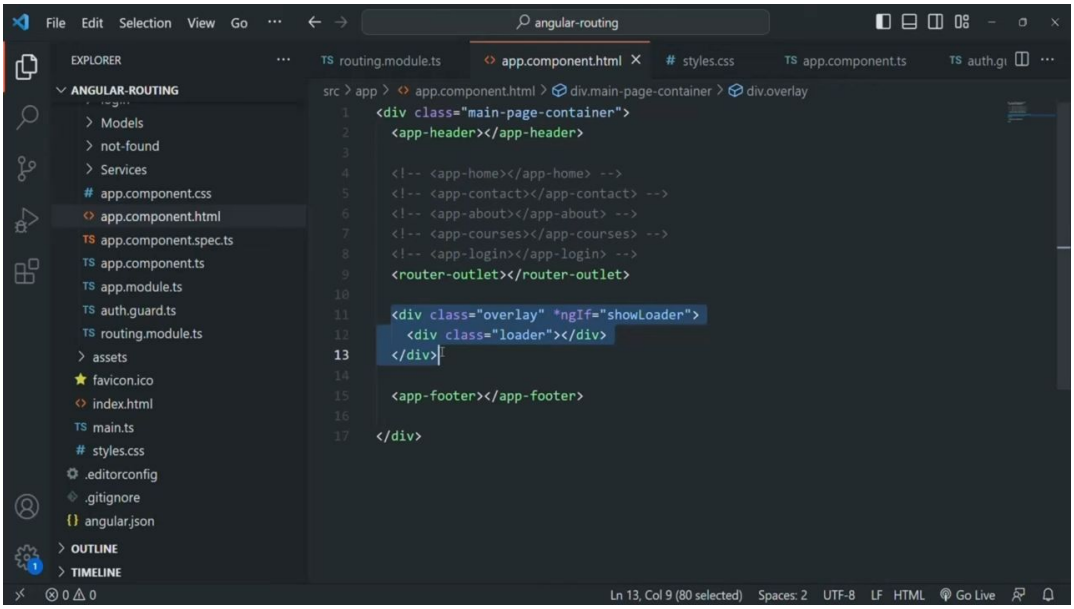
Developers demonstrate how to log navigation events in a web application using the developer console.

5:07:08



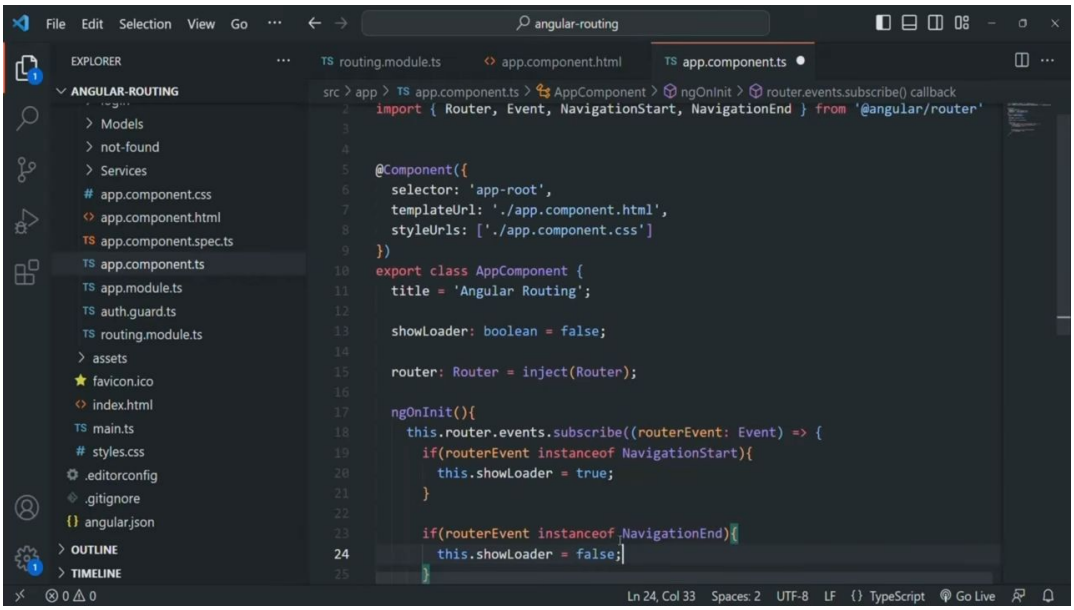
New CSS classes have been added to the style.css file for a loading indicator on web pages.

5:11:56



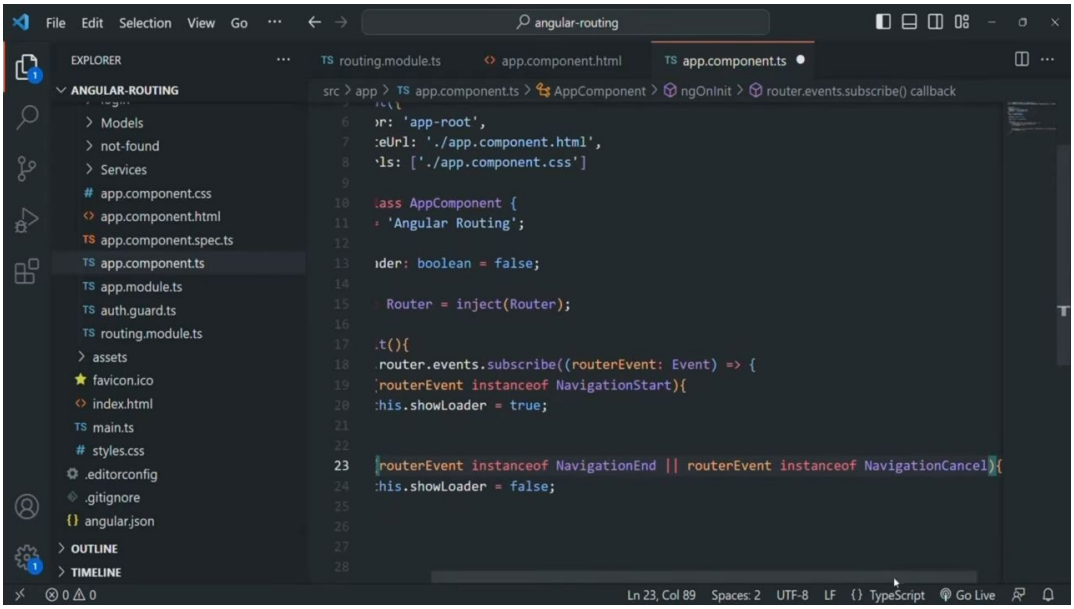
Loading indicator removed from web page based on show loader value in appcomponent.ts file.

5:13:04



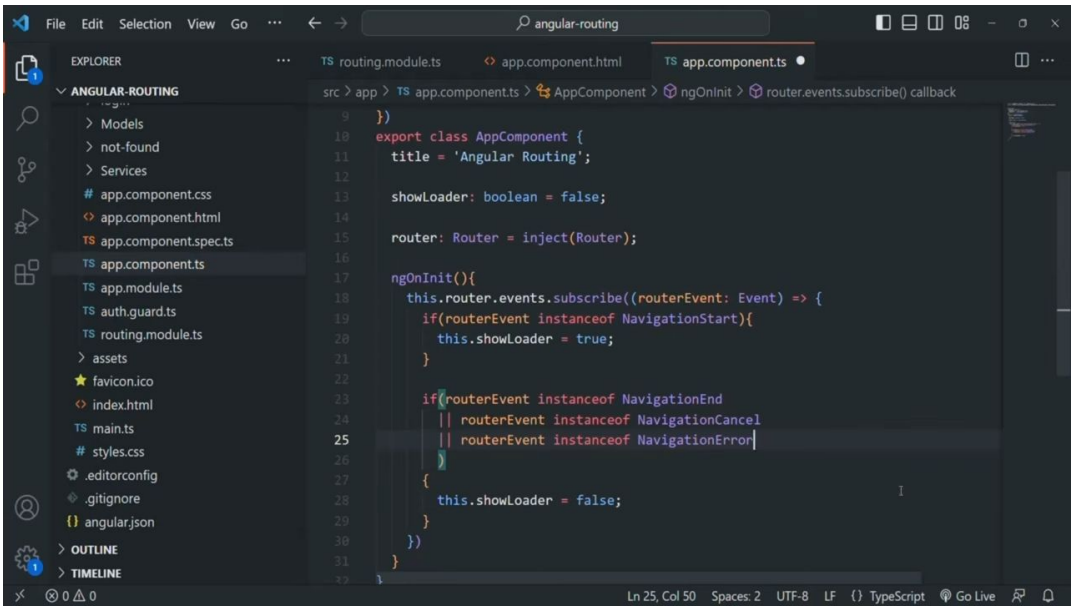
Storing an instance of navigation end and setting the show loader to false can improve user experience on web pages.

5:17:43



To handle the router event and hide the loading indicator, we need to import navigation cancel from angular/router.

5:20:34

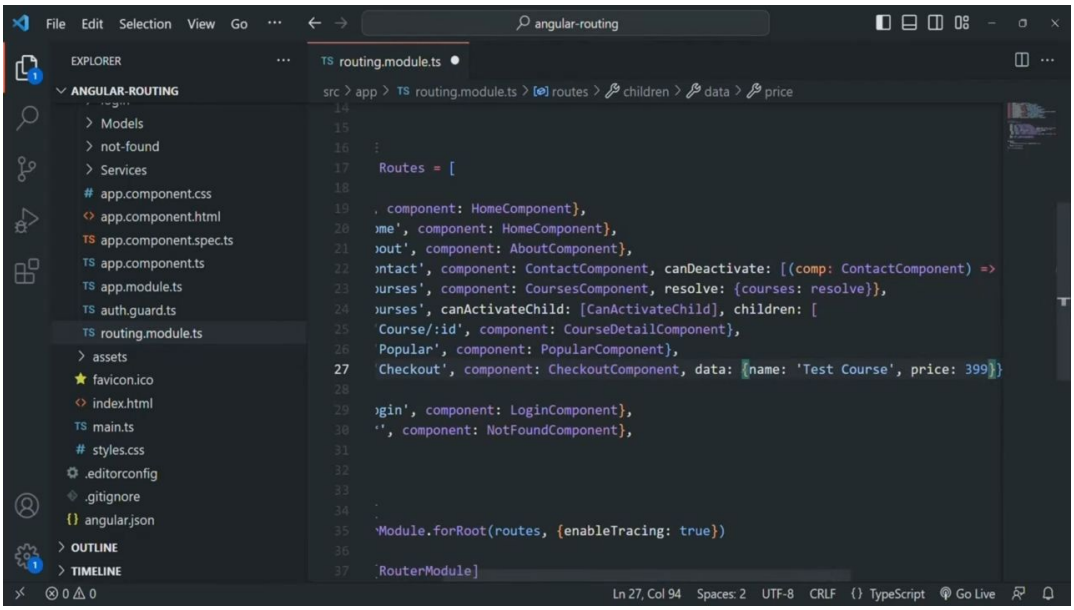


Using router events in Angular to track navigation errors and outcomes.

5:22:03

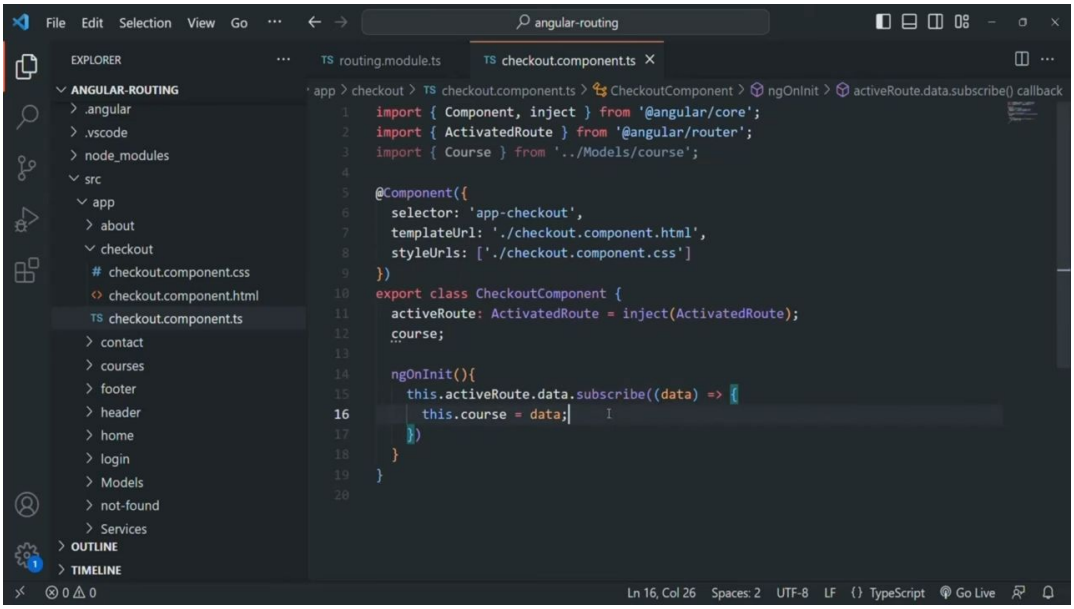
Angular allows for passing data through routes, both statically and dynamically, providing flexibility in application development.

5:23:19

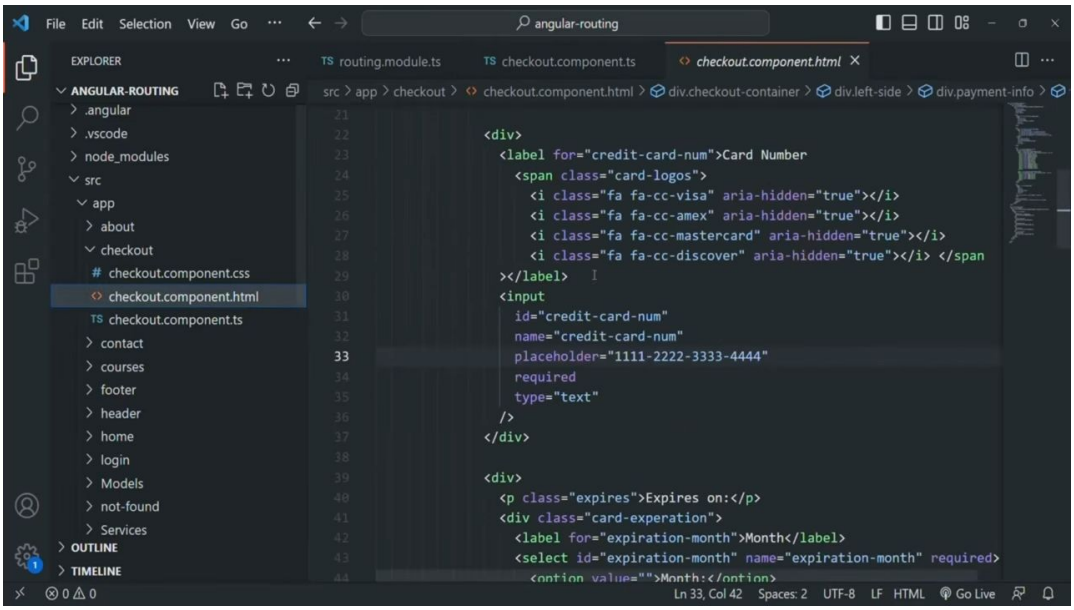


A new course, "Test Course," has been introduced with a price of $399 and will be available for users to navigate to.

5:25:35

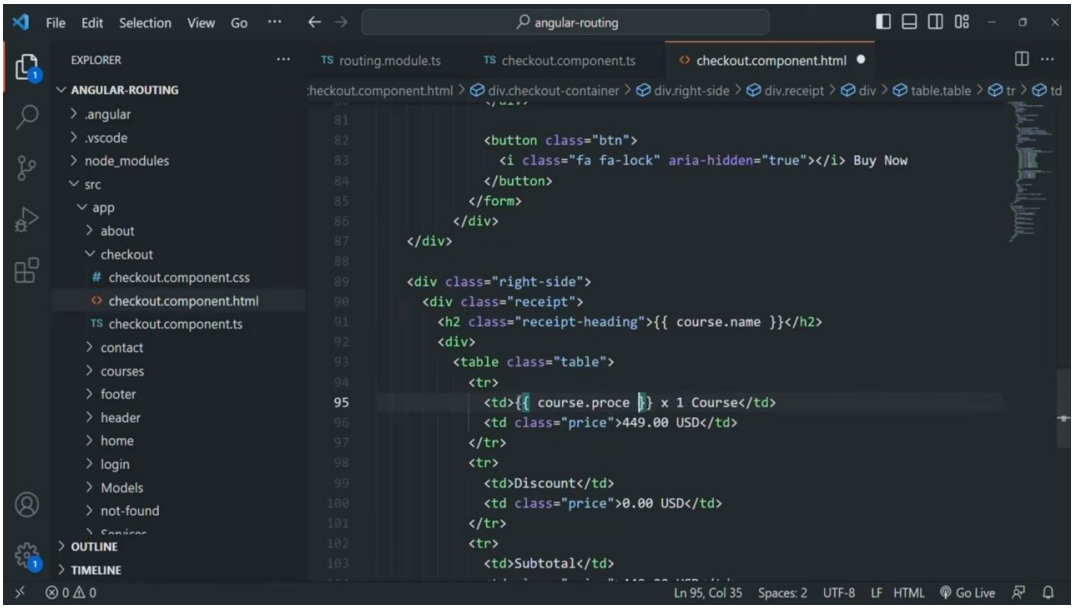


5:28:44



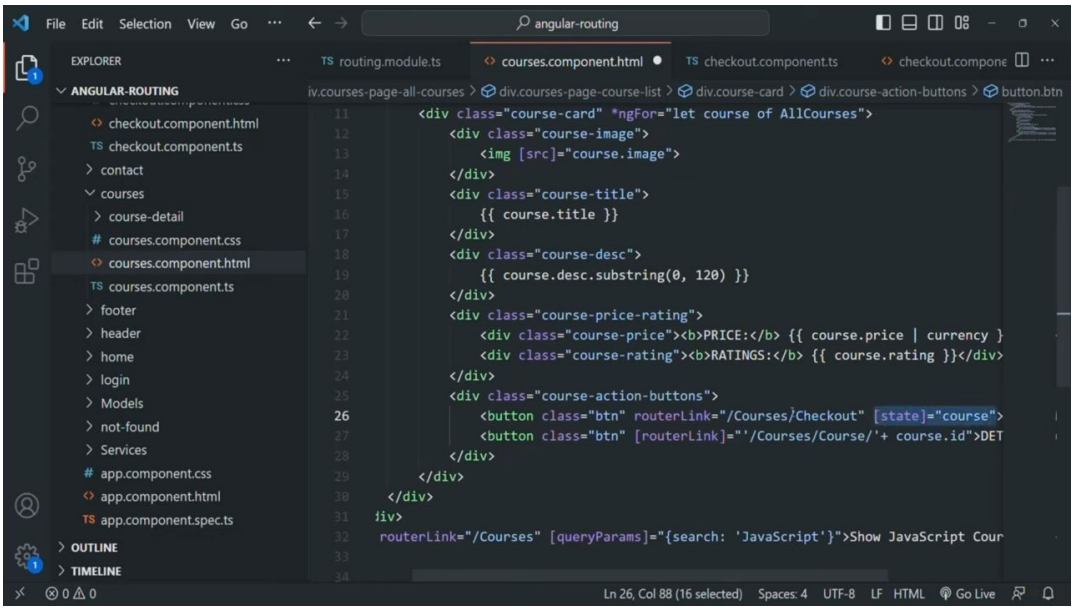
Developers are advised to remove the type in order to avoid errors and use string interpolation syntax for displaying course names in HTML files.

5:28:46



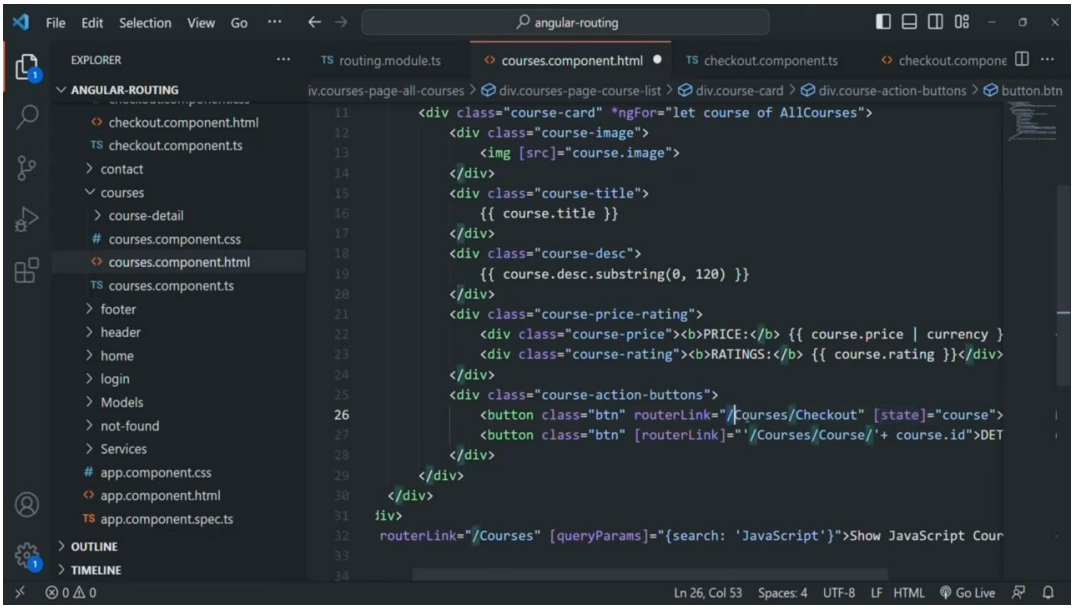
A new feature has been added to display the value of the name property and price in a software application.

5:29:22

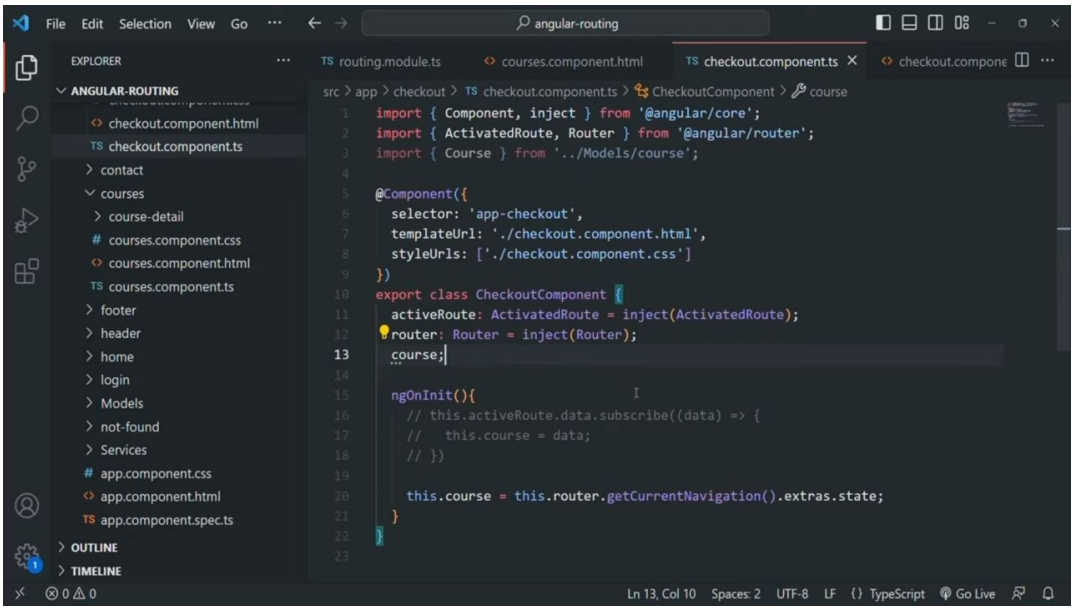


Pass dynamic data with routes by assigning the current course object to the State Property in order to access it in the checkout component view.

5:32:30

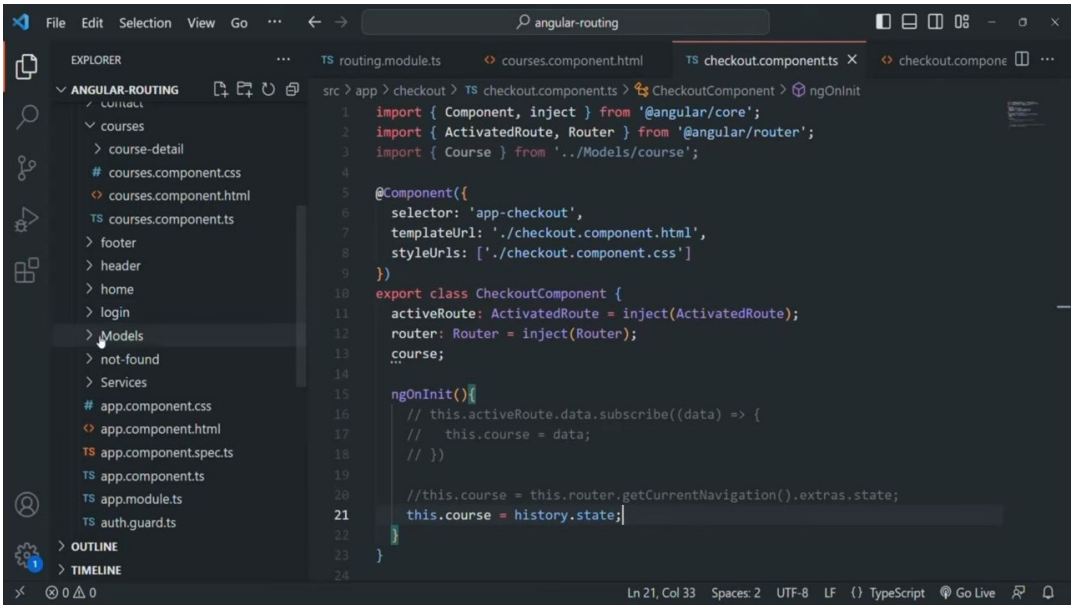


5:32:31



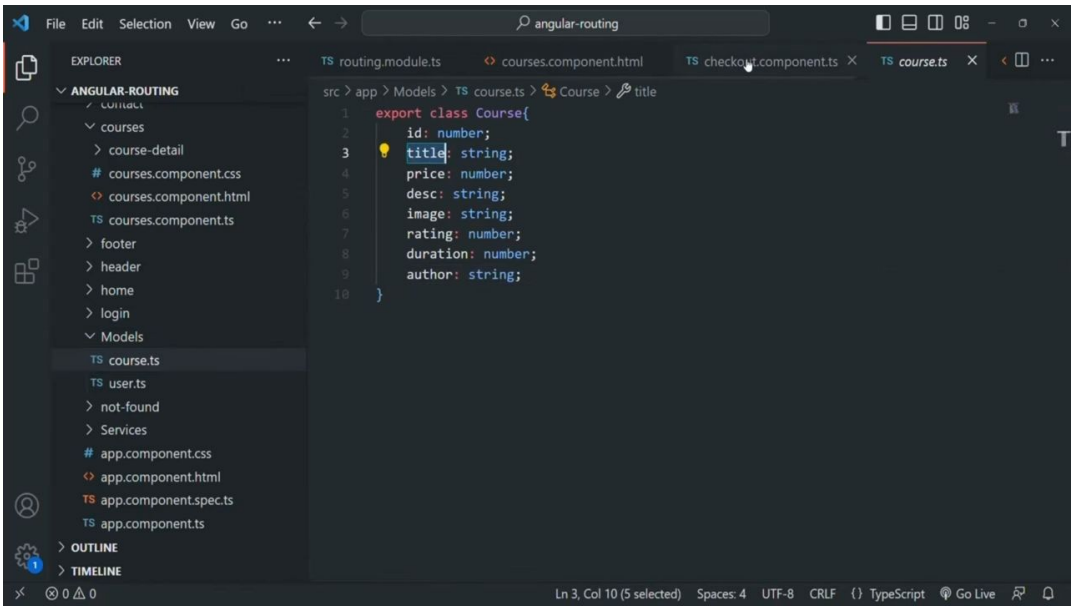
Error in course type causes issues with website functionality, prompting changes to be made.

5:34:51

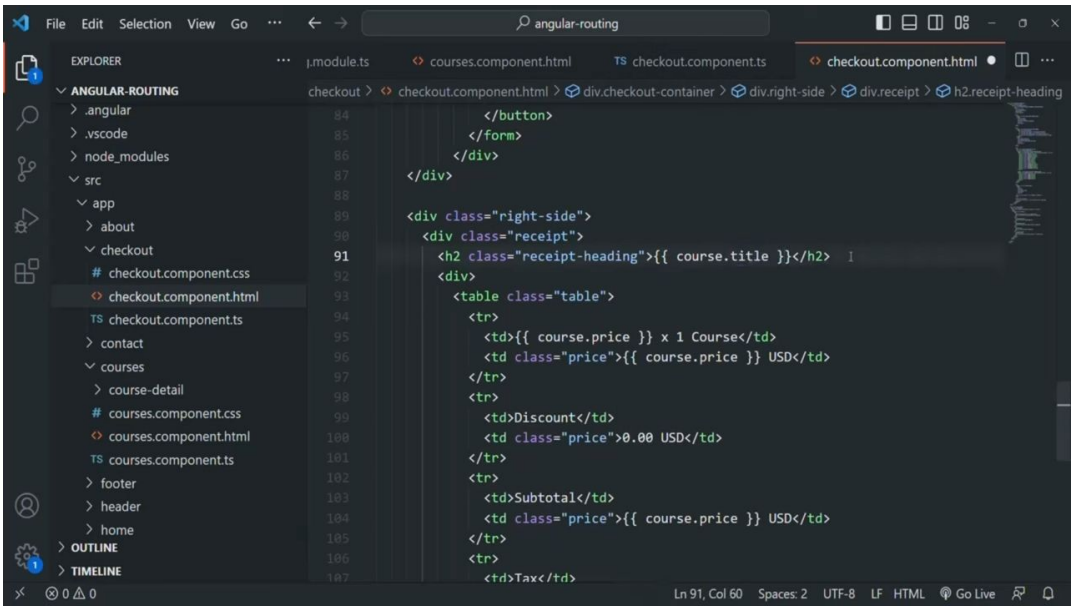


New React course now available with displayed price but missing name property.

5:36:48

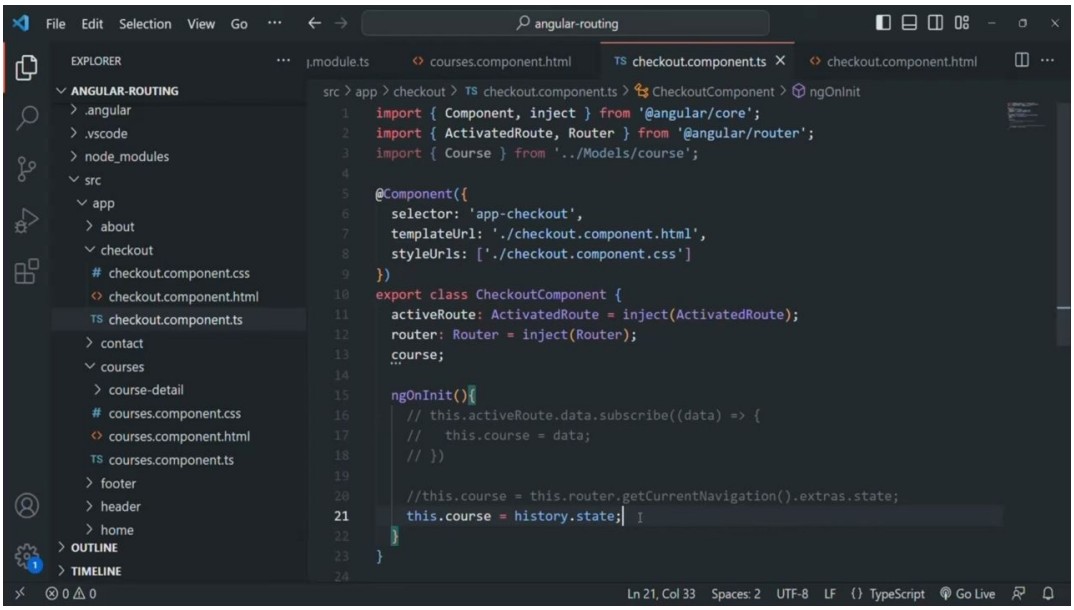


5:36:57



Checkout component.html has been updated to use the title property instead of the name property for a smoother user experience.

5:37:07



Learn how to pass static and dynamic data to a route in Angular and master routing and Route guards with real world examples.

5:38:51