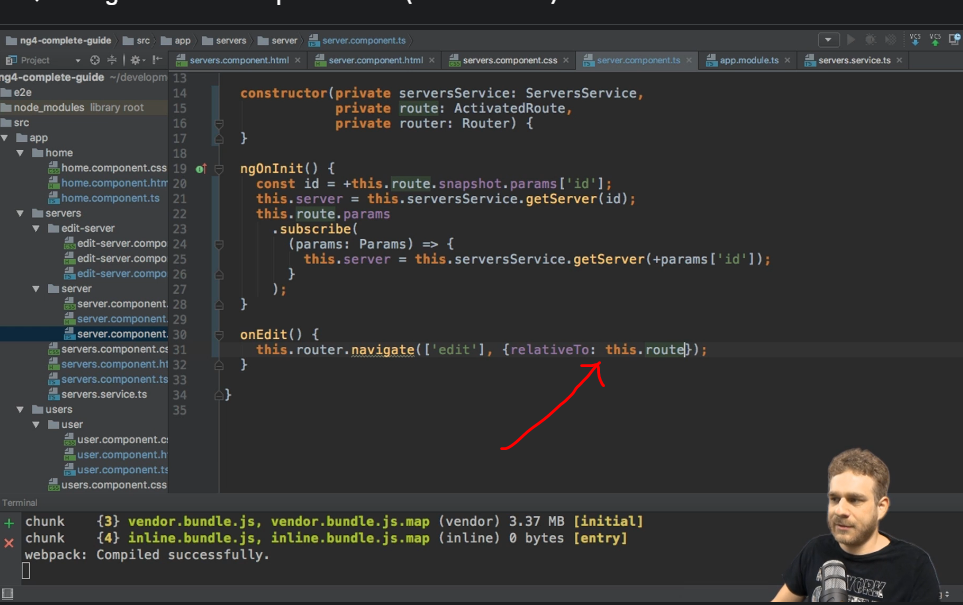
**140. Using Query Parameters – Practice:**

* -: In the last lecture we added child routing.
* Let's now improve this app a little bit more.
* In our servers component we have our links loading the individual servers.
* So the single server component.
* This component here with this allow added query parameter.
* So now in the server component we should add some button where we say edit server because right now we get no way of reaching the edit server component besides manually typing this route here this path here into our URL.
* So let's add a button to the server component.
* I'll add some CSS classes to make this look nice.
* And of course I'll add a click listener where I say onEdit.
* Now again, this is on a single server component.
* So here I'll add the onEdit method.
* And in here I now want to navigate to the edit server component.
* We learned how to do this.
* We need to get access to the router so that we are able to call the navigate method.
* So let's inject it, make sure to also add the import from @angular router.
* And with that injected in onEdit we can navigate, you learned how to do this.
* So maybe a time to pause the video and try it on your own.
* This is how I would do it.
* I would call my router.
* Then of course we have the navigate method and here we need to pass an array.
* And we simply want to append edit to the end of the currently loaded route.
* So you could of course also use /servers the relative path.
* Then use the ID of the currently routed, loaded server again, server ID and then append edit.
* But since we already are on this path why not just use a relative route and just use edit? Now you learned if you want to use a relative route, relative path here when using the navigate method, you need to set up the relativeTo property here, on the second argument on this object, you can pass to navigate and reference the currently loaded route so that the angler router knows to which route you want to navigate relatively.



* So with this, we should load the slash edit route.
* We should append it to the end of our currently loaded route.
* Let's see if this works.
* If we go to servers, click on a server and click on edit.
* We load the edit server page.

Graphical user interface

Description automatically generated

* And now here we want to control on whether the user is allowed to update to edit the server or not.
* The issue is, you see, we theoretically have our query parems here allow edit always set to one in this case.

Graphical user interface, application

Description automatically generated

* But we lose this information as soon as we click edit server.

Graphical user interface

Description automatically generated

* So that is something we need to fix before we can use the information there.
* We will fix it in the next lecture, before doing so let's implement the rest of the logic though.
* Let's say here on the servers component where we set the allow edit query parameter we don't always want to set this to one to true therefore, we want to decide on whether we want to allow the editing depending on the server ID let's say.
* So only if the server ID equals three let's say, only in this case, we want to allow edit so we want to set it to one.
* Otherwise we will set it to zero.
* So that's just ternary expression, checking the server ID.
* And if it's not equal to three, we'll set it to zero.

Text

Description automatically generated

* So we will set the allow edit parameter to zero.
* That's the first step.
* Now an edit server of course, we want to be able to retrieve our query params.
* So here we already get the subscribe method on the query params.
* Now here I want to determine on whether we are allowed to edit or not.
* So I will add a new allow edit property here and set it to false initially.
* But whenever this changes here in the query parameters I want to get my query params, which are alls of type params to be imported from @angular router.
* And then here in the method body I will set this allow edit equal to my query params here, equal to the allow edit ID or property we set there.

Text

Description automatically generated

* And now this ever is one string of one.
* Of course, you could also pass true as a string here.
* In which case I want to set allow edit to true.
* If it is not one, I will set it to false though.
* And in the template of this component we could simply add a heading.
* Let's say where we say, you're not allowed to edit, if, and we can use ngIF for this, if allow edit is false.
* So if not allow edit.
* And we wrap all the rest in a new div.
* So all this part here, only if it is true.

Text

Description automatically generated

* And now all of a sudden we would control this.
* But again, the issue is if we click on edit server, our query params disappear.
* So it is always in the, you are not allowed to edit mode.
* Let's fix it in the next lecture.
* Let's see how we can actually preserve our query params once we navigate again.