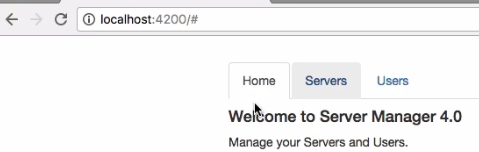
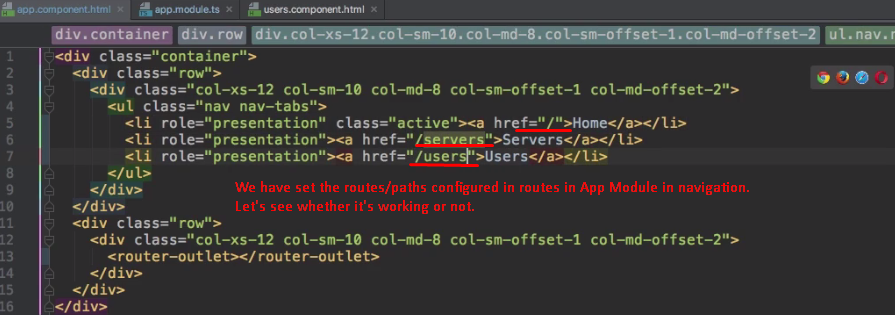
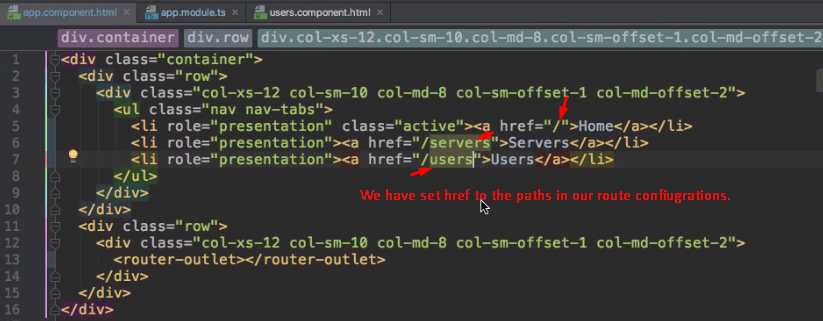
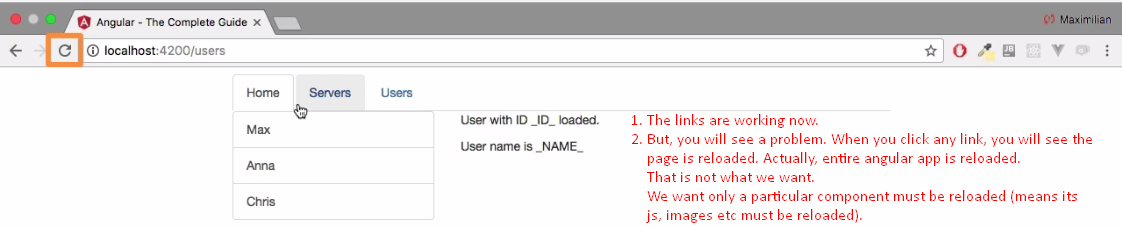
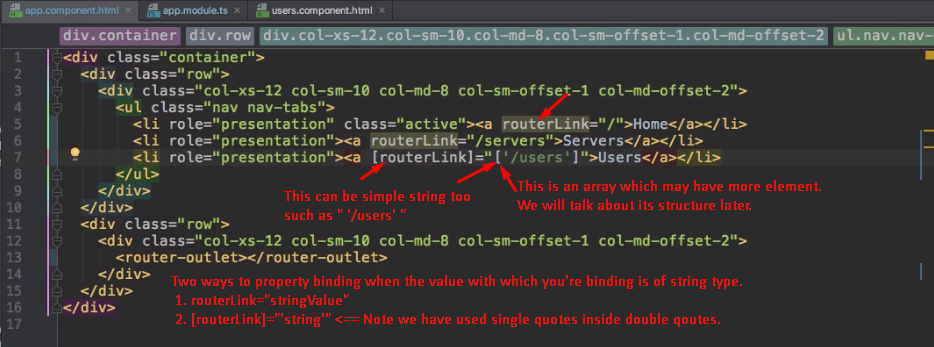
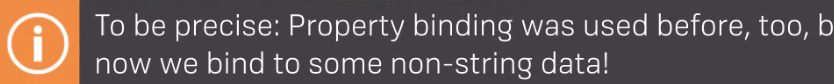
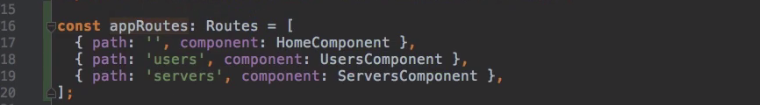
1. In the last lecture, we made our routing work. We added routes and gave to Angular and told it where to load currently active component.
2. **Problem**: We can navigate around by manually typing the path (URL) in the address bar. Not super convenient. While it’s important to be able to load a route directly from the address bar, it would also be to load a route directly from the address bar, it would also be nice to have some working links here in our navigation to have these tabs work. So, let’s add some links to our application.  
   **Requirement**: Navigate by clicking on the links.   
   
3. If we go to our app component, we see here is our navigation.
4. See, in the following slide, we have added all the routes (the **paths** we have set up in our **routes configuration**) to our navigation.
5. 
6. 
7. 
8. The above is working as when we click a link, an http request with a particular URL path is made and angular knows about the **paths are configured with routes**. So, Angular returns the page loaded with the requested component.
9. **Drawbacks of refreshing app**:
   1. Our whole app state is lost as clicking the link refreshes the whole app.
   2. It reloads the app which takes time. 
10. **Solution**: **routerLink directive**:
    1. This can parse a string.
    2. This tells angular that the element <a> on which this **routerLink** is placed will serve as a link in the end but it will handle a click differently as you will see in a second.  
       
    3. **NOTE**: [routerLink]=”’/users’”🡸 Why single quote inside double quote? Because otherwise it will try to find property named /users. So telling it’s not property name but value itself.
    4. **Why array as value of routerLink**? You can pass all segments of your path as elements in this array.
11. Leave this if not clear🡺
12. Now if you see, it doesn’t reload the whole app/page but just the component. Because the router link catches the click event on the element, prevents the default which would be to send a request and instead analyzes what we passed here to the routerLink directive which is path or path of elements (in case of array) and then parses it and checks if it finds a fitting route in our configuration which it of course does because we defined all the paths we’re using here.  
    
13. This is how we can navigate around with a rounterLink and this actually is how we should navigate round because it gives us the better user experience, it doesn’t restart our app, therefore it keeps the app state and it’s much faster than reloading the page all the time.
14. NOTE: Still you can type URL manually in URL bar which would work fine.