1. **Debugging Angular application:**

* -: Sometimes reading error messages is not enough or you don't even get a error message like here, I got my servers here, right? And if I click on them I can delete them and it seems to work until I click the last one
* Or if I deleted all of them, the only one
* So deleting the last server is not working I can click it as often as I want
* It's not going away
* Now if you have a look at our code at the template first we see that on each list item I call on remove server and I do pass I the index of the current duration to that method
* Now, of course, we could simply dive into the TypeScript code here and have a look at this and there we see, well, we get the position of this item and then we simply call splice to remove the item the element in this array at this position
* So you might say, Well, that looks alright
* Sometimes it's just not super easy to debug a message like this because we're not getting a error message but we get a logical error in our app
* Now, debugging is supported by developer tools in a browser though so it would be great if we could simply check at runtime
* Why does this not working? In Chrome, you can simply go to sources and here you will find all these sources imported on this page
* So in this case, all our script bundles there are a couple of bundles as you see
* The important one for us is, of course, the main bundle
* And if we have a look at the main bundle, yeah that's not super easy to debug, right? Now, theoretically, of course we can have a look at it
* And there, let's find this seems to be our template here with container and row and maybe we can find the line responsible for splicing here
* And now maybe we want to debug here
* And if you try to place a break point here, which you, of course, can do in your developer tools you see something changed
* It jumped to the app component, TypeScript file
* Now, remember, TypeScript is not what runs in the browser
* So there the JavaScript, these bundles here are used

**Source Maps:**

* How can we now access TypeScript? Because these JavaScript files support source maps
* Source maps are a little addition
* DCLI kind of adds to our bundles which allow the browser to translate our JavaScript code to TypeScript or to simply map, map the JavaScript code to our TypeScript files
* In development only these source maps will be stripped out for production, of course
* So this is a great feature because now we can place break points here in our TypeScript code, and if I now click somewhere you see it indeed pauses, and now we can check and even see that ID currently is two which makes sense because I clicked this item here which has an idea of two just to show this again Here, now with this item, now it's ID one because, again the middle item, which is ID one in this case because we started zero for the first item and then we see that position is undefined right now
* Now we can move on a step
* Now we see position is two and now we can also check servers
* And in servers we see well position two, of course, is the last server in the list
* So, if you let this continue it looks like the middle one was removed, but actually the last one was removed
* And we can confirm this by checking this
* If I click on the last one which was the thing which was not working we see ID is one, makes sense it's the second element of two, and the ID starts at zero
* So, ID one makes sense, but now position is two because we add one to the ID, so now we try to delete the element at position two in this servers array, and clearly we don't have that element here
* We only have two elements with positions zero and one
* So the logical error in our code is that we add one to the ID here, and that is how we could debug that with the TypeScript code here, with that source map that, of course would allow us to now fix this bug
* Now, correctly you might say, "That's awesome, that's great", but if I have to scan my main bundle to then click somewhere to open the TypeScript file that's not really a great well tool because if that bundle grows bigger it gets even harder to find the right well spot
* Therefore, you can directly access your TypeScript files under Webpack here you should find a couple of sub folders
* And if you open up the dot folder here and then source here in the app folder ends on you find all your TypeScript files
* So here is where you can directly access your TypeScript files in the same structure as in your project to easily find a file you want to dive into and you want to debug with the Chrome developer tools
* So that is another great tool at your disposal besides reading error messages using the debugger with source maps with TypeScript files to either, of course dive into errors for which you have error messages but also into logical errors like here

