**71. Understanding View Encapsulation:**

* -: I'm not sure if you noticed it in the last lectures, with our working application, if I add Servers or blueprints, something changed.
* A new Server down here in the blueprint as well as the paragraph right at the top no longer has a blue color.

Graphical user interface, text, application, chat or text message

Description automatically generated

* Now, it was colored blue before because in the app.
* component.
* css file, we defined that paragraphs should have a blue text color.

Graphical user interface

Description automatically generated

* But both paragraphs were moved.

Graphical user interface, text

Description automatically generated

* They are no longer part of our app.
* component here.

Text

Description automatically generated

* The first paragraph is part of the cockpit, and the second paragraph, well, that is inside of our Server element.

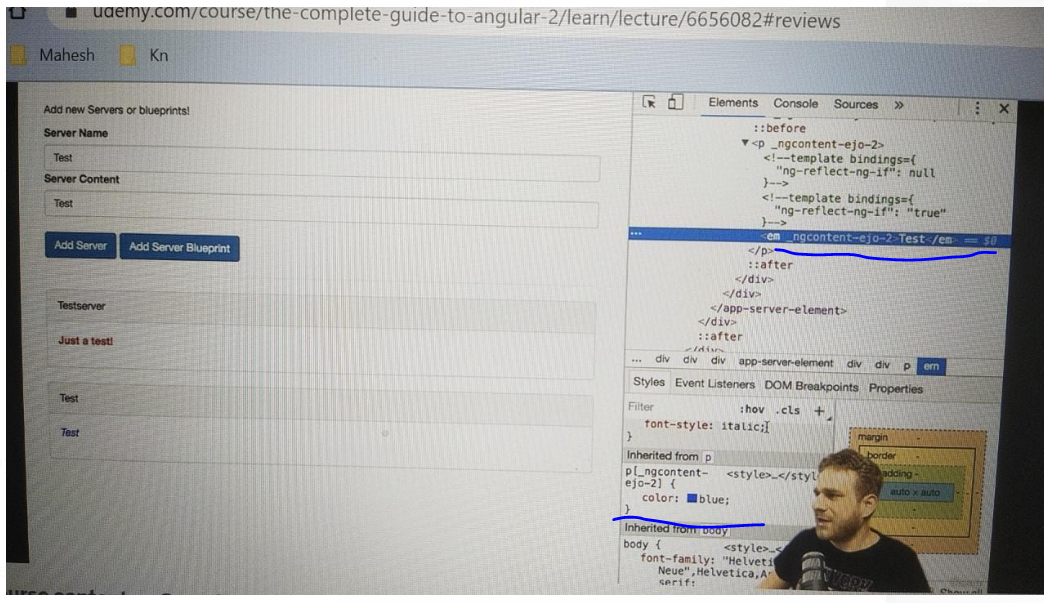
Text

Description automatically generated

* But isn't that strange? Here on this CSS definition, we just say paragraph, so it should affect all paragraphs in our whole app.
* Now, you might say, "Well, no, it's in the app. components.
* css file, so clearly it only belongs to this template.
* " But is that so clear, because the way CSS works, it doesn't really care in which CSS file you define a rule, it simply is applied to the whole document, normally.
* So this actually is a behavior enforced by Angular, which is not the default behavior of the browser.
* *So while of course these CSS files have the goal of encapsulating styles for the component they belong to, this, again, is not a default behavior, Angular gives us this behavior.*

**View Encapsulation:**

* It's a great behavior because, with that, *we can make sure that whichever styles we define in a CSS file will only get applied to the component they belong to.*
* However, here, of course, this means that now we would have to copy this blue color here to all our other components that we want to use it.
* Well, I never wanted to use it on the cockpit anyways, but I will copy it to the server element, and if we now change this and let it reload, and enter something here and add a new blueprint, now you see this is blue again.
* Now, it's interesting if we inspect this text here in the developer tools, you see that, if you look at the styles, clearly here we assign the blue color, but do you see that? The selector changed, before, we had just P here, just paragraph, now it's paragraph and then this very strange attribute, \_ngcontent- ejo-2.



* Well, I don't know about you, but I didn't add it.
* You can see this attribute here on the paragraph.
* That is why the style gets applied here, because this style definition says make every text blue which is in a paragraph where the paragraph has this attribute.
* Something which is true for the paragraph holding this specific text.
* Now, the other paragraph we have in app-cockpit, here in the div, this paragraph does have a different attribute, \_ngcontent-ejo1.
* Down here, we have ejo-2, that is why the styling gets not applied to this paragraph.
* And all these strange attributes, which you can see all over the place here, all these strange attributes are applied to their specific elements by Angular.
* Angular, as I told you, enforces this style encapsulation.
* And it can't do that magically, so the way it does it is it simply gives the same attribute to all elements in a component.
* So all elements in this cockpit component have ngcontent\_ejo-1 attached to it, as you can see.
* And it does this for each component with different unique attribute names, ejo-1 and ejo-0 and ejo-2, and with that, it can make sure that, once it changed your style selector, it automatically adds this attribute selector to all your styles defined for a component, that these styles get only applied to elements of that component.
* This is how it enforces this behavior.
* It kind of emulates the shadow dom.
* The shadow dom is a technology not supported by all browsers where each element has its kind of own shadow dom behind it where you then could assign styles to each element, but as this technology is not supported by all browsers, this is how Angular emulates it, and that is the default behavior of view encapsulation in Angular.