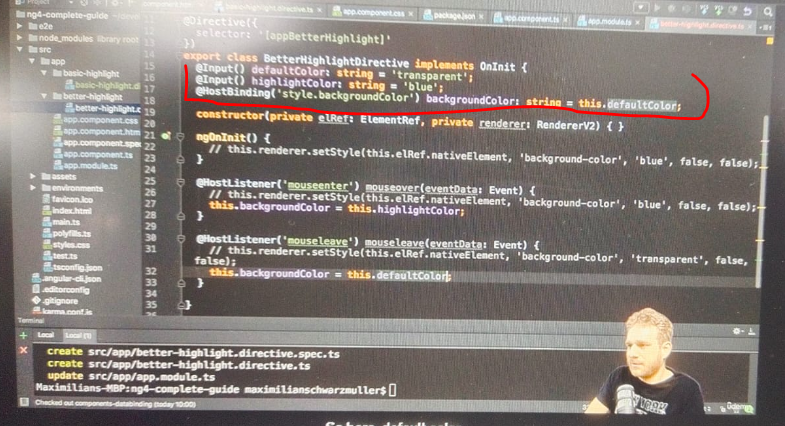
**98. Binding to Directive Properties**

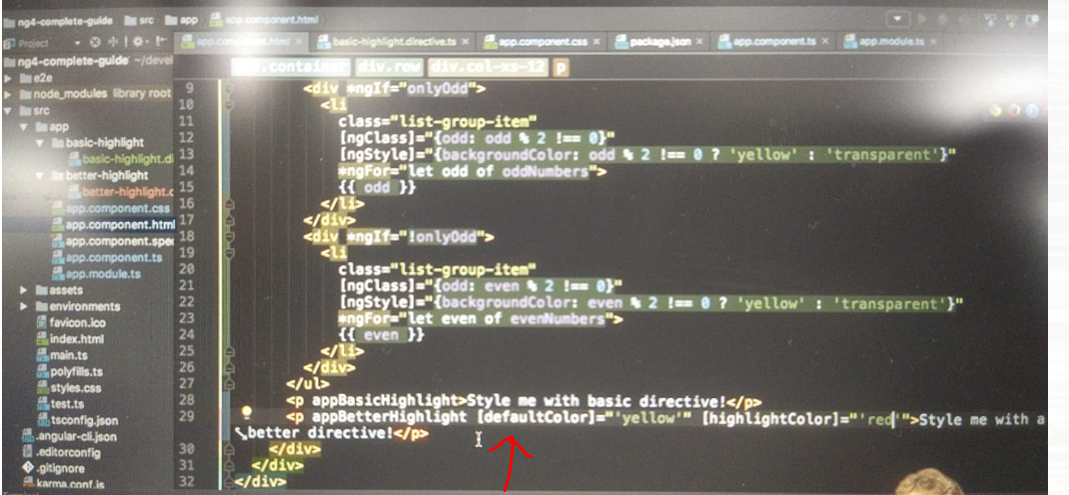
* -: Our directive is looking really great.
* We're almost there but there's one other thing I want to add.
* One functionality I want to add, right now it's dynamic in a way that we can mouse over and then move the mouse away but we can't decide which colors get used.
* *Now if that were a directive, we want to ship with let's say a third party package, we're offering the user the developer using this directive should be able to dynamically set the value.*
* Maybe we even want to do this in our own app because the color we want to set changes depending on some our parameters in our app.
* Right now, the color is hard coded in there transparent by default and blue if mouse over it.
* **Custom property binding**
* So that is something we can improve and we can improve it with a tool we already learned Custom property binding
* As a side note custom event binding also works and directives but you probably won't use that that often.
* So how can we use custom property binding then? Let's add two properties to which we bind.
* I'll add it with input here and I'll name the first one default color and set it equal to a string and even assign a default color, which is transparent.
* Now add input of course needs to be imported from at angular core.
* I'll duplicate this and I'll set this to highlight color.
* That will be blue by default.



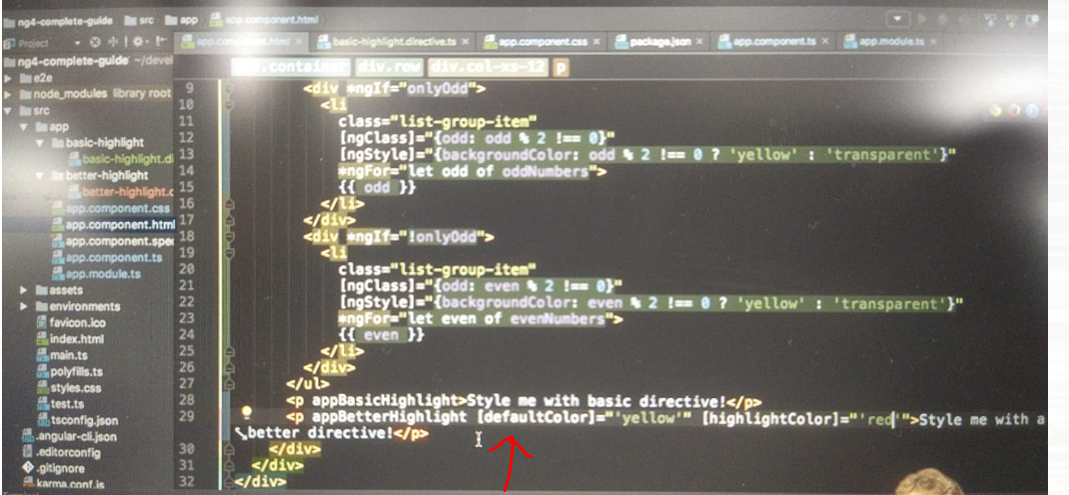
* So we do have some default values to use but they can be overwritten from outside.
* So now by default here I will assign my highlight color for initially setting this.
* And in my host listeners here, if I mouse over it I will assign my highlight color.
* Excuse me here, it should be my default color at the top of course.
* So default color, color when we initialize this and then highlight color once we mouse over this.
* And then default color again once we move the mouse away.
* So here, default color.

**Binding From Outside:**

* So with this it should still work just the way as it did before but now we can actually bind this from outside.



* So in our app component where we use the better highlight directive, we can now bind to default color and maybe set this to yellow.
* Should be a string though yellow.
* And we can bind to highlight color and pass a string, which could be red.
* And now if we save this notice that app better highlight the directives itself.
* The name is not enclosed in square brackets.
* Now you'll see it's red and yellow here.
* Though we also detect a black initially before we mouse over, it's a white.
* The reason for this is when we assign default color here it's just not set here.
* So to prevent this from happening what we can simply do is we initialize it here in ngOnInit stat.
* That's all the before anything has been rendered.
* But after our values here are available so here we can set this background color to this default this default color, and now it should work fine.
* Now we have yellow by default so this now works and now we are able to override this.



**Interesting Things – on way we pass down values:**

* There are a couple of interesting things we can observe regarding the way we pass down these values though.

1. The first interesting thing is that we have two extra directive like looking things on the paragraph and this is just property binding.

b) Now how does angular know if we want to bind to a property of paragraph, which of course doesn't have a default color or to a property of our directive, the answer is it just figures that out.

* ***It simply checks our own directives and so on first before it reaches the custom the native properties of elements.***
* That's another important takeaway.
* *We can bind to properties of our own directives by simply placing them on the same element.*
* These properties, I mean enclosed in square brackets of course.
* Now for NG class you see that somehow the directive itself is enclosed at square brackets and that's a typical use case especially if you only have one property to bind or at least one main property.
* Then you can provide an alias and we can do this here for let's say the highlight color and set this equal to your directive selector.
* So app better highlight in this case if I set this as an alias for the highlight color now this assignment here won't work anymore so I will remove it instead, now I can enclose my main directive into square brackets and set this equal to red.
* So now we should still see yellow and red but now we are having the same style of and closing the directive itself.
* And it's important to understand that this is only an option.
* You can set such a such an alias but that's not something you have to do.
* And by default, as you saw before, the directive name is not enclosed in squared brackets.
* That really only happens if you want to bind to a property which has the same name or alias like your directive selector.
* One other thing about how we pass down data.
* One other thing which is true about property binding in general, if you pass down a string like now we use square brackets and then single quotation marks.
* While you can take a shortcut, you can remove the square brackets and remove the single quotation marks.
* And that is a special case.
* If you are passing down a string, you can do you can add property binding without square brackets.
* If you then also omit these single quotation marks between the double quotation marks and that should work as it did before, no error.
* You can use this, you will see me use this later in the course with official angular directive.
* Be careful that if you use it, it's really clear that this is property binding, that no one thinks that this could be a real attribute existing for the element you placed it on.
* Then this is absolutely fine to be used.
* And this is our directive finished with some extra features like host listener, host binding, and dynamically setting values from outside.