**100. Building a Structural Directive**

* -: Now that we understood what the role of the star is we can write our own structural directive.

**Create new Directive: unless**

* So let's create a new directive and I will use the CLI for this.
* I will name it unless, so I will basically create the opposite of the NGF directive.

**Building Structural Directive:**

* This directive here will attach something only if the condition is false, NGF does it if the condition is true.
* So here I'll again delete the spec file here and in the unless directive I get my selector, which is fine.

1. **Adding input and set Unless**

* Now here I need to get the condition as an input.
* So I'll add, add input.
* And remember in the end we will use property binding with the square brackets because Angular transforms it for us.
* We need to of course import input from @angular/core.
* And then here I want to bind to a property named unless which kind of simplistic condition we get.
* But whenever this condition changes so whenever some input parameter here changes I want to execute a method and therefore I can implement a setter with the set keyword.
* This now turns this into a method though technically and that's important to understand, this still is a property, it's just a setter of the property which is a method which gets executed whenever the property changes and it of course changes whenever it changes outside of this directive.
* So whenever the condition we pass changes or some parameter of this condition.
* Unless therefore needs to receive the value the property would normally get as an input.
* And we know that this will be a boolean because it will be our condition in the end.
* So we could also name this, condition.
* Then we can check if the condition is not true which is the case in which I want to display something because unless is the opposite of NGF and if the condition is true well then I want to display nothing.
* So that is how we get the condition, how we use it how do we display something.
* Keep in mind that our UnlessDirective here in the end will sit on such a ng template component because that is what it gets transformed to by Angular if we use the star.
* So we can get access to this template and we also need to get access to the place in the document where we want to render it.
* Both can be injected.

1. **TemplateRef injection via constructor**

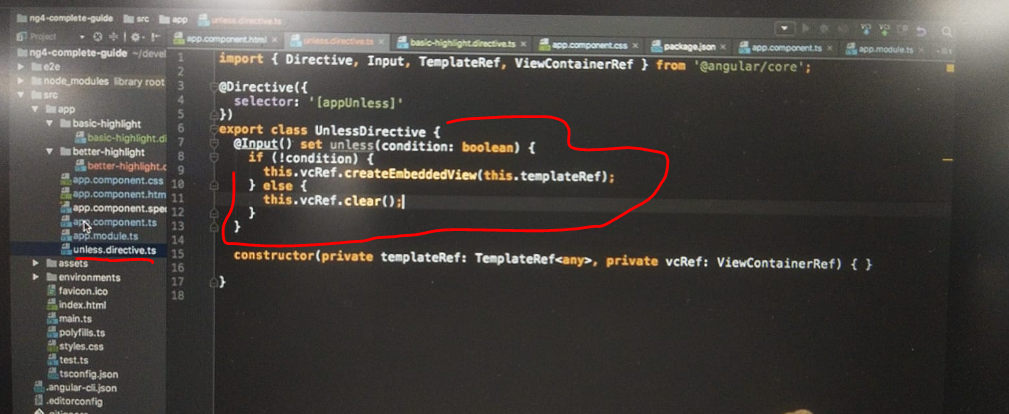
* The template can be injected by adding template ref any name you like.
* But the type is of type templateRef.
* So just like elementRef gave us access to the elemented directive was on, templateRef does the same for a template and this is a generic type.
* You can simply pass any here and we need to import templateRef from @angular/core.

1. **ViewContainerReference injection via constructor**

* The second information piece we need is the view container.
* So where should we render it? The template is the what, now the question's where? So I'll name it a vcRef for ViewContainerReference.
* And the type is ViewContainerReference or ViewContainerRef which is also imported from @Angular/core.
* *That marks the place where we place this directive in the document. Angular marks this place. And you can see this if you inspect it in the developer tools actually.*

1. **Call createEmeddedView() of viewContainer in set Unless:**

* So with these two tools available we can use the vcRef whenever the condition changes to call the createEmbeddedView method which that's just what the name implies it creates a view in this view container.
* And the view simply is our templateRef.



* So this template we created there is exactly, this reference to the template there is exactly what we need.
* Well, and if the condition is true in this case, so if it's not what we are looking for then we'll simply call the clear method to remove everything from this place in the DOM.
* With that, our own directive is set up.

1. **Adding UnlessDirective as part of Declarations in app.module.ts:**

* Of course we need to make sure that it is added here.
* The CLI did this for us.

1. **Using our own directive in app component:**

* And now in the app component we can use our own directive maybe to replace NGF here.

Text

Description automatically generated

* So I'll comment out all this stuff here and only copy the original div down there.
* Comment this in of course.
* So here, instead of using ngf, I'll use appUnless and the star is important because it still is a structural directive otherwise we would have to manually write it with this ng-template.
* Syntax you'll learn before.
* So here of course we don't want to check if onlyOdd is false because keep in mind the unless directive will check for the opposite already.
* So here we have to pass just onlyOdd.
* And with this in place we get that we can't bind to appUnless because it's not a known property.

**Error:**

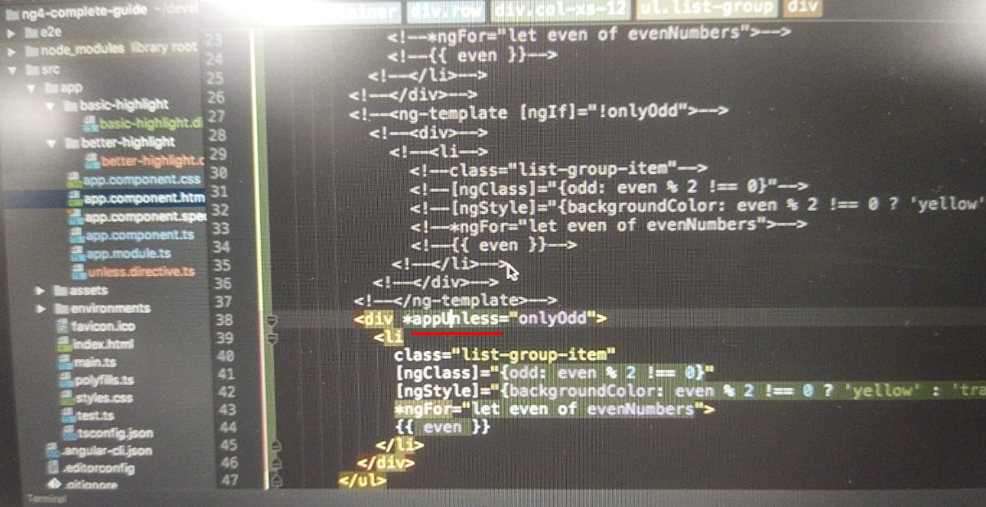
A picture containing text

Description automatically generated

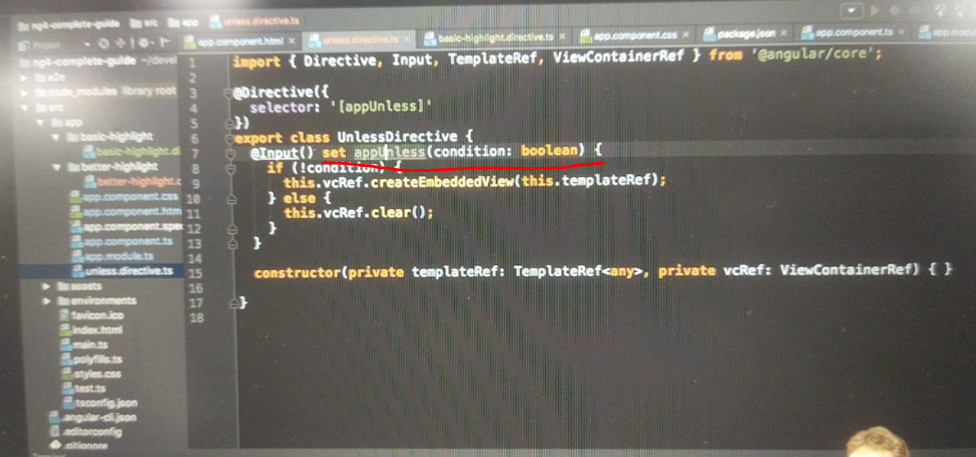
* Why do we get this? Can be really hard to track.

**Reason for Error:**

* We get this error because what we are trying to do is we have property binding here, custom property binding with @input and we're binding a property named unless.



* Now keep in mind the star automatically transforms this in this ng-template syntax, where we then try to property bind to the directive name, which is appUnless.



* So we have to make sure that our property here shares the name of the directive, appUnless exactly the same, the same as the selector.
* Now with this, it works fine and as you can see, if I toggle here we get the same behavior as before even though I commented out the NGF block and use my own appUnless directive instead.
* So this is our own custom structural directive build.