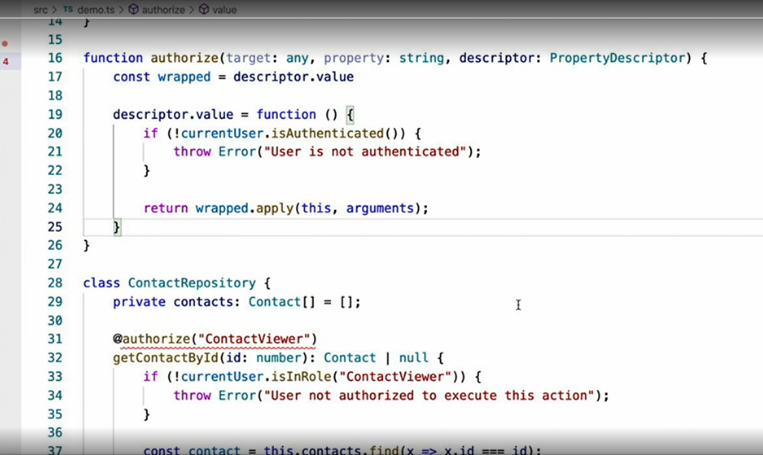
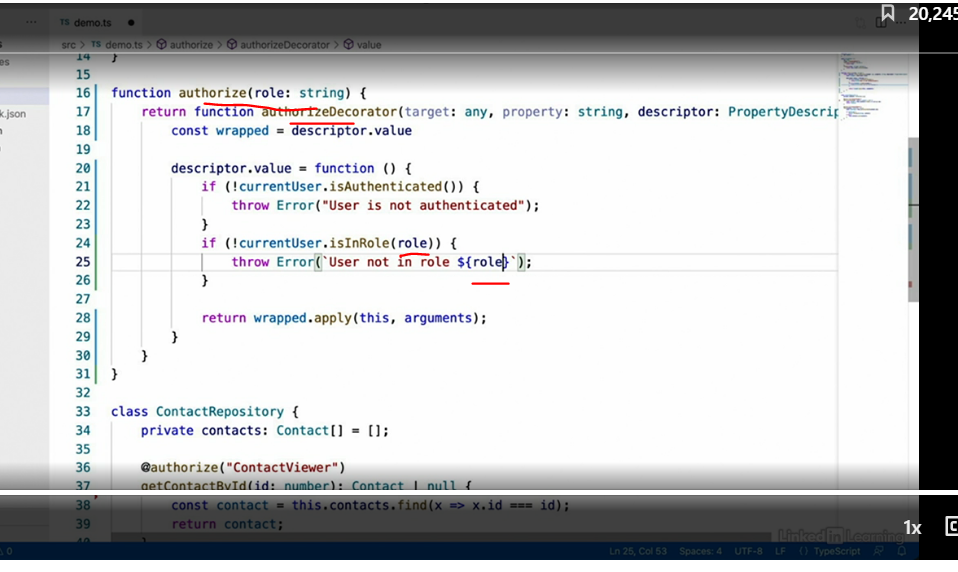
* - [Instructor] In the previous video, I showed you how to create a method decorator to wrap a methods functionality.
* And I used this authorized decorator to demonstrate.



* I also pointed out, however, that this implementation didn't allow us to pass in any parameters, such as which role we expect the user to have, like the existing code does.
* *The problem is that a decorator must have the signature that my example decorator currently has with the target property and descriptor parameters.*
* How then do we pass in a custom parameter like I want to do here? **By wrapping the decorator definition in another function called a decorator factory.**
* As its name implies, a decorator factory ***is a function that creates decorators.***
* In other words, our current decorator definition now becomes the return value of another function, like this.
* With this wrapper in place, we are now free to define whatever signature we like, including whichever custom parameters we want.
* For example, I can now add the name of the user role that I want to ensure the user belongs to.
* The decorator function within this factory can then access these parameters in its logic like this.



* With this decorator and factory in place, that code that I showed in the beginning will now work just fine, accepting the role that I'd like to check for, or allowing me to add any other custom parameters in logic I might want to add in the future.