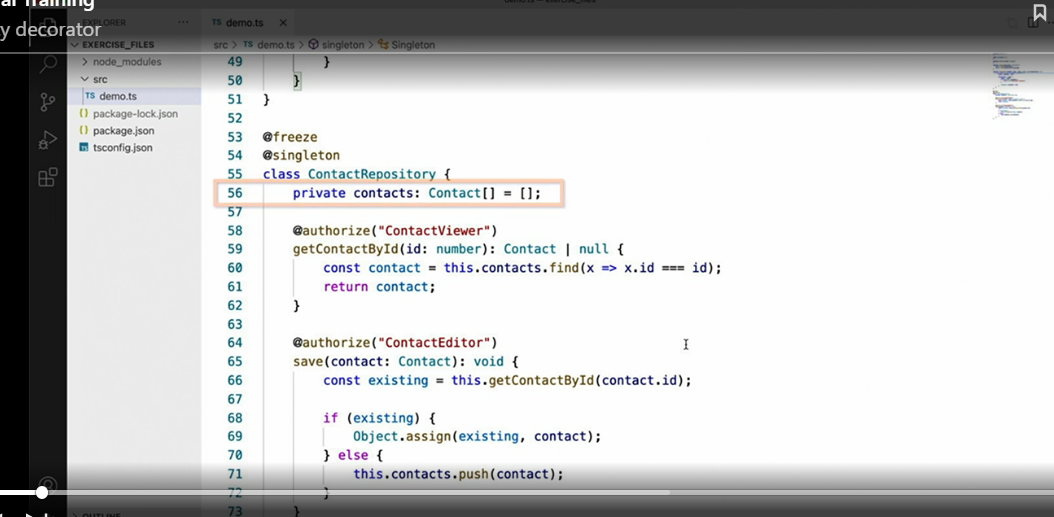
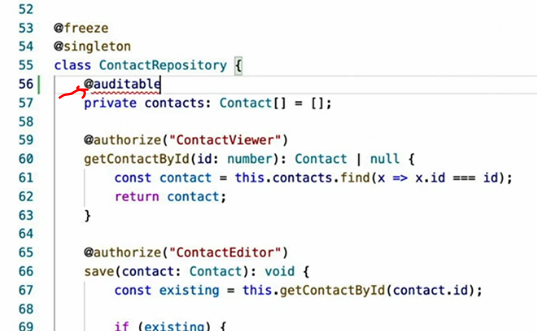
* - [Instructor] So far in this chapter, I've shown you how to create method decorators and class decorators.
* In this video, I'm going to show you about the third kind of decorator**, the property** decorator.
* As its name implies, ***property decorators are decorators that are applied to properties of a class,*** like the contacts property of our contact repository example, and you apply a decorator to a property the same way you apply one to a class or a method like this.



* Here, I'm applying a decorator called auditable, which I'll create in a minute, to the contacts property.



* *The implementation of this decorator will log a message to the console anytime the value of this property has changed.*
* ***Since methods are just properties of a class that happen to be functions****, the definition of a property decorator looks very much like a method decorator, a function that accepts two parameters,* ***the target and the property name.***
* And just like the method decorator, these two parameters correspond to the instance of the target object that is being decorated and the name of the property being decorated.
* Now that I've defined the decorator function, let's implement it.
* To do so, I'm going to use the object.
* **define property API** to easily define getter and setter functionality on this property.



* The short explanation of this code is that first, I save the initial value of the property on line 55, ***then overwrite the property with a custom getter and setter using the object.***
* define property starting on line 58.
* In the custom setter on line 60, I log a message to the console on line 61 anytime the property value is changed.
* Unfortunately, a more in-depth explanation of the define property API and its pros and cons is outside of the scope of this course, which is, quite frankly, why I saved this video for last.
* Most of the property decorators I make end up looking almost exactly like this.
* However, I highly encourage you to get a solid understanding of JavaScript properties and the define property API before trying to do anything too advanced in a property decorator.
* Despite this warning, however, I do still encourage you to use property decorators to make your properties far more dynamic than just saving and retrieving their values.