So earlier in this section, we created this FakeFileReader, and we used this to write one unit test for our video service class. So in this test, we're testing only one of the execution paths, in ReadVideoTitle method, that's the execution path that we are testing. But what if you want to test the other execution paths? Well, we need to change our FakeFileReader, so instead of returning an empty string, we should return a real video object, serialized as json. But here's the problem, if you modify this code, the existing unit test is going to break.

So we should create another FakeFileReader that implements this interface and in that fake object, instead of returning an empty string,

we should return a real json object. We can see that this approach is very time consuming. Creating all these fake objects by hand takes a lot of time.

So that's why we use a mocking isolation framework. So there are frameworks out there that help us dynamically create these fake, or mock

objects. So we don't have to hard code them by hand. We can create them dynamically as part of our tests. and more interestingly, we can program them to behave any way we want. So we can program them to throw exceptions, to return values, to raise events, and so on. Again, we have different isolation or mocking frameworks out there, we have mock with a q, we have Nsubstitute, we have FakeItEasy, we have Rhino Mocks and so on. After all these frameworks the one I personally prefer is mock. So in the next lecture, I'm going to show you how to use this framework to dynamically create or mock objects.