Earlier in this section, I told you that good unit tests, should not be too specific or too general.

In this lecture, I'm going to show you examples of specific and general tests. So here in the fundamentals folder, we have this class HtmlFormatter.

So a simple method, FormatAsBold takes a string and simply wraps it with a strong element. You want to write a test for this method. So,

in our unit testing project, I'm going to add a new class, HtmlFormatterTests.

Let's add Test

Fixture attribute here. Correct the test method. public void.

FormatAsBoldWhenCalled ShouldEnclosetheString WithStrongElement.

So, let's create a new instance of this class. Formatter is a new HtmlFormatter, so we call FormatAsBold, give it a message, like ABC.

A simple string, we don't have to make it complicated, we get the result, now we need to make an Assertion. I'm going to show you different ways to write assertions here, some of these Assertions are more specific, some are more general, and then I will give you guidelines about what is a better solution in this case. So, here is an example of a Specific Assertion. Assert That result, Is Equal to we have this strong element, we have abc and then /strong. This Assertion is specific, because we are verifying the exact string that we should get from this method. In this particular case, this is a good solution, because what comes out of this method is important, we want to make sure that message isproperly formatted as bold. But sometimes your method might return an error message. You don't want to write specific assertions there, because it is possible that you might modify that error message in the future.

You may add a period, you may add an exclamation mark, you may add

or remove a few words, if you're test is specific and is looking for the exact same error message, every time you change that error message that test is going to break. That's why I said your tests should not be too specific. In contrast, they should not be too general, because if they are too general, they may pass all the time even if your production code has an issue. So let me show you a more general way of writing an Assertion here.

More general, this time we want to Assert That this result starts

with strong element. So, instead of using the Is class, we're going to use Does. Again here we have a few static helper methods, so Does StartWith strong.

So imagine, this is the only Assertion I have in this test method. All this method is looking for is the presence of strong at the beginning

of the string that we get from this method. It is possible that we have a bug here, so instead of returning this string, we return just this. That test will easily pass. Because it's too general, right? Now, we can make this a little bit more specific, so we can duplicate this Assertion and also check that this string ends with /strong.

Now, if you return strong, that test is gonna fail.

So, we should make sure to add strong at the end as well. But we still have a bug here, because we have not included this content here. So back to our test method, we can add another Assertion that this result Does Contain abc.

So here's the takeaway, when testing strings, its better if your Assertions are a little bit more general, because if they are too specific, your tests can break easily, However, in this particular case, I'm going to go for the first solution, so I want to make sure that this is exactly what we get from this method, otherwise, we may end up with a bad HTML document. But if this method was returning some kind of error message, I could use these methods to start with end with, and Contain, to look for specific words in that error message. One last thing, when you are writing Assertions against strings, by default these Assertions are case sensitive, if you want to

disable that, you can use the ignore case property. So here, after calling the equal to method, we can chain this property IgnoreCase.

We can also use the same technique when writing more general Assertions. So we can say results should start with strong and we don't care about the case sensitivity.