Now as this code gets longer, it doesn't really feel right to keep it inside the loading screen anymore.

So let's refactor it and try to move our location getting code into a separate class.

And I'd like you to do that as a challenge.

I'd like you to refactor this code so that all the logic of getting the current location will be handled by a dedicated location object.

This means creating a separate location class in the location.dart file.

This class needs to have two properties, a latitude and longitude.

It also needs to have a method could getCurrentLocation.

This is where you'll need to move the try catch block from the loading screen.

Back in the loading screen, update the getLocation method so that you create a new location object. You call the get current location method and then you print the value stored inside the latitude and longitude.

This challenge will test your understanding of async and await.

So keep those principles in mind.

I'll give you a few seconds to pause the video before I walk you through the solution.

OK, ready?

Here's the solution.

The first step is creating a new class in the location.dart file called Location. And this location class is going to have two properties.

It's going to hold onto a latitude and also a longitude.

Now they're both going to be doubles because that's the output from our geolocator when we do position.latitude or position.longitude, the data type will be a double.

So now let's go ahead and create a class method and we'll call it getCurrentLocation and this is not gonna take any inputs, but it is going to use that geolocator package.

So let's go ahead and cut our try catch block out of our get location method and instead paste it into this getCurrentLocation method. We're again gonna use the geolocator to get the current position.

Now of course we need that await keyword,so we have to mark this as async so that we can await for the current position to be found before we return it as the output of this method. And instead of printing the position, we're going to assign the position.latitude and position.longitude to these two properties.

So we're gonna say latitude is now gonna be equal to position.latitude and longitude is gonna be now equal to position.longitude.

So now when we call getCurrentLocation, it should try to wait to get the current location and then assign those values to our latitude and longitude.

Going back to our loading screen inside getLocation, we should be able to tap into that file by importing our services folder and then we're going to search for our location.dart file,so that's the full path to this file. And then we're going to create a new location object which we'll just call location and we're gonna set it to equal a new location object.

And once we've initialized that location object, we're going to say location.getCurrentLocation.

Now remember, if we want to use the result of the location.latitude or location.longitude which comes from this location object, we have to wait for this getCurrentLocation to complete.

So ideally, we want to be able to mark this as await. But remember that we can only wait on methods that return futures.

So over here instead of just returning a void, let's return a future void instead.

And this way if we head back to our loading screen, you can see we can now wait on this to complete before we start tapping into the location.latitude or location.longitude. And it's only by waiting for this method to complete can we reliably actually get access to the latitude and longitude.

So now let's hit run and see our code in action. So you can see we're able to tap into the latitude and longitude in our loading\_screen.dart even though the location code is actually all located inside a separate file, the location.dart file.

Now asynchronous programming is notoriously hard in any programming language not just Dart.

So if this was still confusing, I recommend heading back into the lesson we covered async in a way and just watching it again, maybe on double speed, just to make sure that you're revisiting those concepts again.

It might take a couple of tries before all the pieces start coming together and it starts making sense.

But if all of that was easy, then we're going to continue ahead to making API calls and actually using these latitude and longitude to get the weather for our current location.

So for all of that and more, I'll see on the next lesson.

