So in our app at the moment, we're able to get the temperature from the weather data. But all the rest of it is actually still just hard coded text.

If we take a look at our weather.dart file, you can see that in here we've included some code for you just so that you don't have to type out all of these condition checks and add a whole bunch of else ifs one by one. What we've got here is we've created a class called WeatherModel and it has two methods at the moment.

One which is called getWeatherIcon which takes the condition as an input so that's going to be an integer. And we're going to take that condition and check it against the whole bunch of things.

So if we have a look at the API documentation for Open Weather Map, you can see that there's a section on list of condition codes. And it tells you that the condition code that you get back through that weather ID can be matched against a table to determine what the weather condition is.

So say if we got a 300 code, then that means light intensity drizzle and then as that number goes up, the drizzle gets heavier and heavier until we get to shower and then finally rain in the 500 range. So we've looked at that table and compiled a set of if else statements that checks against that conditional code to return a corresponding emoji, which is what we're going to display inside this text widget instead of that static sunshine emoji.

And there's also a getMessage method which takes temperature as an input, checks against that temperature and returns a message.

So for example if the temperature is really warm above 25 degrees Celsius, we'll return 'It's ice cream time'.

Or if it's between 20 and 25, will say 'It's time for shorts and T-shirt'. And you can customize these messages if you wish or you could change these to Fahrenheit depending on what you prefer.

But essentially, this one will give us a weather icon and this one will give us a message and they're both going to return Strings.

So inside our location screen, let's tap into that file by importing our weather.dart file right at the top.

So we're going to import and we're gonna start the path from the services folder.

We're going to look for the weather.dart file. And again right up here just above where we've created our temperature condition and city name variables,we're going to create a new one which is going to be of type WeatherModel and I'm going to call it simply just the name weather and it's going to be equal to a new weatherModel object.

Now we can use it inside our updateUI and we're going to pass over this condition and we're going to call that weather.getWeatherIcon method,we're gonna pass in this condition object.

So now if we got back a condition of 300, then it's going to check against all of these if and else statements and it'll return the correct emoji depending on that number. The icon that we get back from this method is what we want to use inside our scaffold, inside our build method.

So instead of saving a condition variable, I'm going to change that to a string type and I'm going to call it our weatherIcon.

So now I can change condition to a variable that's only going to exist inside updateUI and then I'm going to save the result of this method call to our weatherIcon.

So now we are creating a object that will only exist in here based on the weatherData.

So we're pulling out that weather condition and then we're passing the result of that into getWeather

Icon as the condition.

Now if you really want to cut down on your code, you could potentially replace that with that, and that would still work.

But I think this is a little bit more readable and I find it easier to understand what's going on just by splitting it into two lines,but it's up to you. So now that we've got this weather icon that's being stored in this variable, we can now use it inside our text widget down here. And we can replace that the sun emoji with our weatherIcon.

Now emojis are processed exactly the same way as strings.

So it's just as if we had a string and we can parse them around using that data type.

Now here's a challenge for you. Use the weatherModel object to get the message parsing in the temperature that we get inside updateUI.

So we know that we already get a temperature from our weatherData and we've already converted it to an integer.

So it's now time to use this method to try and get back the message. And we're going to fill that message into here. And then the second part here is going to be the city name, and the city name of course comes from our weather data.

So here is where it's going to come from.

And in the end it should still read a message in a particular city name.

So pause the video and see if you can complete this challenge.

All right.

So we've got temperature, weather icon and city name properties.

So we now need to add another one which is called weather message and we're going to work out the weather message by using our weather data model,so our weather object, and we're going to call get message. And then we're going to pass in the temperature which is going to be this one because it is expecting an integer, so it will pass in that as the input.

And now our weather message can be used inside here instead of this piece of text.

Now notice how this piece of text has double quotes around it, even though normally we should be using single quotes.

And the reason for this is because we have an apostrophe in the middle of the sentence.

So even though Dart convention normally says that you should have single quotes around all of your text and whenever you have another single quote inside, you can escape it using the backslash.

And we've seen this and done this quite a few times. Now in the wild, you might see people using this as a shorthand trick. If you enclose a string that contains a single quote within a double quote, then you won't actually have to escape that single quote.

But now that we're going to use a variable, it no longer matters what we use in there.

So let's change it back to the Dart default the single quote and we're going to replace this part of the message with our weather message property,so it's going to start out with that dollar sign. And then the second part is going to be from our city name. So we can replace that part with the city name property.

So we're going to add another dollar sign, we're going to add a city name and you can keep the exclamation mark or take it out depending on how enthusiastic you feel.

Now all we have to remember is that if we're using properties to update the state of our widgets, let's say that we've got some new weather data, then we have to remember that all widgets are depending on these properties to figure out what they need to display.

So when this updates, if we want our widgets to reflect the update, we of course have to wrap it inside a set state.

So let's go ahead and put that around all of those property changes. And now we can hit save and run our app.

So now you can see that all the data that's onscreen is updated based on the live weather data.

So at the moment in Cupertino as I'm recording this, it's 12 degrees centigrade and it's sunny.

And the message is telling us to bring a jacket just in case in Cupertino.