Now if you're somebody who's quite sensitive to design and user interface styles, then you would have noticed that there are no such things as dropdown lists on

iOS. You almost never see this unless the designers really screwed up.

So this is very much a Android style user interface element. On iOS the way that we would select from different options is not a dropdown list but instead it's a picker view.

So it's something that scrolls and we can use the scroll behavior to choose a new selection.

So how can we implement this in our app? When we're using the widgets from Flutter,so far we've limited ourselves to material components things such as the sliders and buttons and the things that we've use so far exists both on iOS and Android. And they'll look good on apps which are run on both platforms.

But for some things such as the picker view, it's a very iOS specific user interface element.

So if we take a look at inside this set of Cupertino or iOS style widgets, then the Flutter team has already thought about this for you.

If you wanted certain things to display in your iOS app which are iOS specific for example the Cupertino picker or the Cupertino switch or things such as the Cupertino style navigation bar, then we can use this particular library of widgets,the Cupertino widgets.

So let's go ahead and change out our code to instead of using the dropdown button, to use a Cupertino style picker.

So I'm going to select everything that contains my dropdown button,

I'm going to cut it out and I'm going to put it aside for later.

So I'm gonna comment it out of the code as well.

And now I want to provide a child to my container which is a picker. So to be able to use the widgets from the Cupertino catalogue we have to import the Cupertino package.

So if you search for Cupertino, you'll find packages:flutter/cupertino.dart

And now we can use that widget we saw here which is the Cupertino picker.

And in order to create it, we simply create a new Cupertino picker widget just like every other widget we've used.

So let's put that in here.

So Cupertino picker requires three properties to be filled.

The item extent which is the height of each of the items inside the picker, the on selected item changed,so this is what should happened when the user scrolls through that wheel and changes the selection,and finally the children what's going to be displayed inside the picker.

So let's go ahead and fill these things out now.

The item extent I'm going to set to 32 pixels because I think to my eye it looks quite natural.

Feel free to make it taller or shorter if you want to. Now the on selected item changed is going to take a callback similar to our dropdown button. And when the user changes the picker, then it's going to scroll through the wheel and once they've settled on one of those items, then it's going to trigger this callback.

So it will give you the selected index in the picker.

So if it fell on the first line of the picker, that will be zero.

If it fell on the on the third line of the picker, it will be two etc. And then we'll be able to use this callback to access what the user selected.

So let's print it out for now, selected index into the console.

And finally for the children property, we have to specify what we want to display in that picker. If we take a look at this children property, you can see that it's actually got a very simple type that it wants.

It's just a list of plain widgets.

It's not like picker widgets or the drop down menu item widgets.

It just wants a simple list of widgets. So we can very simply just add a list of text widgets into here.

So let's use a text widget and let's call this one USD and then I'm going to copy it a few times just as when we started out for the material dropdown button.

So I'm going to add euros and also pounds. And now I've provided all three required properties.

So let's hit save and if we take a look inside our app, you can see we now have a picker. We can scroll through and pick the item that we want.

And once we've settled on an item, you can see we get the index of the item being shown.

So if I scroll up a bit, you can see we get the first item and then if I scroll even further we get the zeroth item. That is all because our on selected item changed is getting triggered and the picker is passing over the value that the user selected into our callback where we are printing it into the console.

So in terms of the color scheme, I think it will look nice if it matches that blue background color which is a light blue color. So I can change the Cupertino picker's background color property to that same color which is colors.lightBlue, let's hit save.

So now it matches with everything else and it blends into one piece.

Now it's time to populate the picker with all the currencies from a list of currencies with a FOR loop.

And I'd like you to complete this as a challenge. Similar to what we did for the dropdown items instead of creating this list manually,

I want you to create a new method up here.

So it's going to be called getPickerItems and it's going to return a list of widgets which we are going to put into our picker as it's children right here. So we're going to replace this hard coded list and of course we're going to be using the currencies from our currencies list just as we did before.

And if successful, you should have a picker that should be able to scroll through all of those currencies in the list.

So pause the video and try to complete this challenge.

All right.

So as we did before we had to use a FOR loop to loop through our currency's list for simplicity's sake.

And again I'm going to use a for in loop and I'm going to create a string that's going to be called currency and it's going to loop through each currency in our currencies list. And inside the loop, I'm going to create a new text widget which is going to have the currency in the text widget.

Now I can create a new list of text widgets if you really want to be specific, and we could call this pickerItems and we can set it to equal a empty list.

Now it's really important that you don't just initialize it as null.

Because if we have a null list and we tried to add to it, say pickerItems.add this isn't going to work.

And it's actually going to crash because we can't add to a null list.

We can however add to an empty list which is just a new list constructed with nothing in sight.

So this is a really important point to note for both lists and maps.

This can cause some bugs if you're not careful. Now that we have a empty picker items which is a list of text widgets, we can now add to it and we're going to add that text widget with the currency inside. Our loop is going to go through every currency in our currencies list, insert it into a text widget and add that text widget to our list of text widgets. And then we're going to return that picker items list as the output of our method.

Now it's important that you don't return it inside the FOR loop,so inside these curly braces right here, because that way you're going to return the first text widget prematurely and you're not letting the FOR loop loop through all of the currencies in the currencies list.

So now you can also change this to be more specific saying that our output is going to be a list of text widgets.

Now the reason I can do this is because a text widget is a subclass of stateless widget which is a subclass of widget. And you can see this outlined under the inheritance section in the Flutter documentation.

And now we're ready to call this method in our build method to it right here.

We can delete this hardcoded list of children and instead we're going to call get picker items which is going to slot into here.

So now let's save and let's check out our picker and you can see we can now scroll all the way from Australian dollars to Canadian dollars to all of the currencies that we have in our list. As you can see we can use Cupertino widgets from the Cupertino package as easily as we've been using material widgets.

It's just a case of adding it to our build method and initializing it with various properties that we can set. Now in the next lesson we're going to talk about how can we have both in existence? How can we get the Cupertino widget to show

up when the user is running our app on iOS and to get the material widget to show up when a user is running an app on Android?

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