All right.

So our slider is looking beautiful and it's working beautifully as well.

But for the keen eyed amongst you especially those who are designers, you will have noticed that there is a major difference between our slider and their slider.

So one of the things is that they obviously have a white active side and a gray inactive side which is very easy to change because we can simply tap into that property, acticeColor, and change it to colors

.white.

But as soon as you do that you'll notice that the thumb color,so the color of the main toggle, also changes to the same color as the active track which is probably not what you want.

Also if we look at this one, that holder or handle is much much bigger than what we have here which is a tiny one.

So how do we make our design look more like this?

Well if we go over to the documentation on the slider class, you can see that we have our usual properties which we change say the activeColor or the inactiveColor.

And once you get to the bottom of the list, you'll notice that there's nothing that allows us to just change for example the size of that handle or the color of that handle without affecting this active track here.

So what can we do beyond changing the properties of the slider?

Well here's one way of customizing a Flutter widget to be more in line with exactly the design that you want.

And if we read through this, you can see that to determine how the slider should be displayed, a slider uses the SliderThemeData available either from a SliderTheme widget or the ThemeData, a widget above it in the widget tree. Or you can override some of the colors but fine-grained control comes from the SliderThemeData.

This is one of the things that we can do and we talked about themes before.

So we already added a primary color to our app theme and we changed the scaffoldBackgroundColor across our app using this ThemeData. But we also said that we can have specific themes just for individual components.

We can do that by embedding widgets inside another widget.

So in this case we could embed it inside a SliderTheme and a SliderTheme has a property called data which is expecting something called a SliderThemeData.

And once you hit ENTER, you can see that there's loads of things that you can set for example, the track height or the activeTrackColor the inactiveTrackColor, the thumbColor, all of these things.

But in order to use a SlideThemeData like this, you would have to provide a value for everything because otherwise it will default to null, which is no color or no trackHeight.

And if we had save right now, you can see that our slider is completely broken because none of these properties can be null,we had to provide a value. But what do you do if you're kind of happy with the default theme but you just want to tweak a couple of things without having to provide like 20 values and type out all this code?

Well there's something that's really helpful across all of the widgets that Flutter has and it's something called .of.

So we know that we already have a default SliderTheme. And we can tap into it and use a method called.of which will return the data,so that data property, from the closest SliderTheme in the given context. So we can write SliderTheme.of.

And then we could provide a build context which comes from right over here.

And this context is basically the current state of our app.

It's how our app appears as it is.

And that will include the current SliderTheme.

So then if we say .of, we get a copy of what it looks like by default as it is. And then we can write again using that .copyWith which we saw previously where we copied over our dark theme and simply added a couple of changes.

Now inside this copyWith, we can provide only the things that we want to change about the SliderTheme.

And we don't have to create the entire thing from scratch.

So the things that we want to change are for example the thumbShape.

So instead of having it as the default size, we can change its size by saying that we want a round slider thumb shape which has a normal radius of 6 but we now want a larger radius.

We're going to change it to maybe 15.

And now you can see when I hit save, my thumb is a lot bigger. But notice that when I normally click on it, it has a little overlay a little transparent outer ring.

And that's now gone because it's smaller than the normal fully opaque shape.

Let's go ahead and add that and make it bigger as well. So we can add our overlay shape to be a round slider overlay shape.

And by default, notice that it's 16.

So just only one pixel bigger than our thumb shape.

So let's make it a lot larger.

Let's make it 30.

And now when I click on this, you'll see my overlay animate and show up. It's looking pretty good now.

What about our thumbColor?

Well we can change that using the thumbColor property, and we're going to change it to our previous pink color which is a custom color that is going to be 0xFFEB1555.

And that's that pink color that we have pretty much throughout our app.

So let's hit save but our slider is not changing and it's because our activeColor is overriding that thumbColor.

Let's get rid of it.

So let's add instead an activeTrackColor, which is going to be colors.white.

And now if we save, you'll notice that the left hand side is white, the right hand side is still gray and our thumb is pink. But when we click on it, it doesn't actually have an overlay color.

So let's add a overlay color as well.

And that's going to be the same color that we have for our thumbColor but we want it to have a little bit of a transparency.

Now here's where the first part of my color comes in.

Do you remember I mentioned that there's four components to the code here and it always starts out with 0x and then it's two characters for the alpha or the transparency amount,and then it's red green and blue? What if we wanted to make it more transparent?

Well we would modify these first two numbers.

Now I quite liked how much transparency I was getting in the beginning when I had the default amount enabled.

Now how can I find out what the default amount was?

Well I could go into my slider theme by holding down my COMMAND on Mac or CONTROL on Windows, and I can click on the SliderTheme which takes me to my SliderTheme widget.

And here I can search for that overlayColor property and you can see that if we go through all of them,you can see that the color is set as the primary color of the theme with a alpha amount that is something called an overlayLightAlpha.

So let's try and find this overlayLightAlpha.

So I've highlighted it and hit COMMAND + F or CONTROL + F and we're going to go through this document and you'll find it as a constant in here.

So they've set the overlayLightAlpha as 0x29.

So that gives us about 16% opacity they say.

So now if we copy that over to our color and we replace the first four characters with 0x29,and we hit save, then you'll see that our thumb now has a overlay color of 16% opacity of that pinkish color that we've been using. And this now looks fully like the design that we see over here. So as I mentioned before, we can either keep our slider theme here or if you had multiple sliders in your app and you wanted a app wide slider theme, then you could copy your slider theme and paste it in here. And that would mean that all of the sliders across your app will have the same theme.

But in our case, we only have one slider so it doesn't really matter where we have it.

And I prefer having everything related to the same component all together in the same place. And to just keep it consistent, our inactiveTrackColor should probably also go in here.

So we'll add the inactive track color and we'll paste this color over into here and we can delete it from here.

So that means all of our styling is done in our theme and all of the properties or functionality is done inside the actual slider component.

Now you can of course extract this widget if you wanted to use it in multiple places, but in our case we only really have one.

So I'm just going to keep it in as it is. And also if you wish to,you can move all of these colors and themes into a constant and you could keep this main file a little bit cleaner if you wanted to. In this lesson,what I'm trying to show you is that anything is possible. Any design that you can dream up or your designer can dream up is fully possible either by creating your own widgets that combine different Flutter widgets,or whenever you're using Flutter widgets such as material components, you can always tap into the theme to be able to customize it further than just the properties that it normally comes with.

So for example if you are using a alert dialog that comes from the material component, then you can take a look in the theme class then you see that there's a dialog theme which you can change. Or if you're using an app bar, you can change the app bar theme or the text theme or an icon theme or a tab bar theme.

Basically all of these material components can be further fine tuned using the theme data.

Now what if that is not enough? Because at the moment, we're using a slider because it looks very similar to the design that we're trying to achieve.

So why reinvent the wheel right?

Why not just customize it?

But what if it's not enough?

What if we wanted to create something that is really different? Well we can also build our very own widgets from the very basic Flutter components such as basic shapes like circles or squares. And then change it's elevation or change its background color and change how it behaves, completely define our own widgets and the way it should work.

And we're going to be exploring exactly that in the next lesson.

So all of that and more, I'll see you there.

