Now you can see that even though all the cards look the same which is how we're able to use our reusable card to style each of them and have them laid out using our expanded widgets, but their content are all different. And usually in Flutter when we're using the widgets that Flutter have created, they tend to address this by having a child property.

So for example the expanded widget will have a child or the container will have a child.

And this allows us to reuse the same widget but to add custom content for each of these widget that we use.

So let's go ahead and create a new child property so that we can do the same with our custom reusable

Card. First things first,we're going to create our custom child property and we're gonna call it cardChild.

Also this has to be final because our widget is immutable.

And then this is going to have the type of widget and it's going to be called cardChild.

Well there's a new property but we now have to initialize it because we're getting an error that tells us the final variable car child must be initialized.

So we do that in here.

Now I don't think the child property should be required.

So we don't need to add the required annotation. All we need to do is just write this.cardChild.

So that means if we create a new reusableCard, then we can if we wish to add a child to go inside our widget.

But if we don't need to have any content, then we can simply just have it as is. Now the final thing is where is this child gonna be placed inside our widget tree? Well it's going to be a child of our container.

So we can tap into the container's child property and we're gonna say that whatever cardChild gets passed into this reusableCard will be the child of our container, and that's it.

That's all we need to do.

Now that we've updated our reusableCard, then it's time to actually add some content.

And the first one that we're going to address are these cards that will allow the user to choose their gender.

So we've got a icon which comes from a font awesome Flutter package.

And then we've got a little bit of text underneath just as a description.

So let's get started by customizing our reusableCards at the top row and get the styling down first before we code up any of the functionality.

So first things first, we need to get some icons from font awesome.

We're going to need the font\_awesome\_flutter package.

So let's go ahead and add this to our pubspec.yaml which at the moment doesn't really contain any custom packages.

So right below where we have cupertino icons, we're going to add in font\_awesome\_flutter. And I'm just going to paste that in here. And then we're going to add the obligatory YAML markup which is going to include a colon here and I'm going to add a caret sign up here as well.

And this way we show that we're happy with it using any major versions of font\_awesome\_flutter that are 8.something.something but we're not going to go beyond eight to nine.

So now that we've updated our YAML, we're going to click on packages.get and we're going to import the package into our import page.

So we're going to import the font awesome package and then we're going to be able to use it inside our reusableCard. So first things first, we're going to target our first card here which is going to look something like this.

So we'll have everything centered in the middle of the card and we're going to have our Mars icon from font awesome and a little bit of text below.

So this is probably easily achieved using a column as a child.

So let's go ahead and add that cardChild which we created earlier on as a property, and the cardChild is going to be a column which is going to have two children. And those two children are going to be firstly a icon which is going to be created from a font awesome icon. And the one that we need is the one called mars, which is generally the male symbol. Now that we have the icon in,

I'm going to increase the size of the icon to 80, just make it a lot larger than how it normally is.

And then I'm going to add a little bit of a gap between the icon and the text. And to do that we usually use a sizedBox. And we can provide a custom height property to specify how much distance we want between the icon and the text.

And in this case I'm going to pick 15. And now finally we can add a text widget and it's just going to say the word male in all caps. And then I'm just going to add a little bit of style which is going to be a text style widget and all I'm going to do is increase the font size to 18 and also change the font color to a custom color and it's going to be 0xFF, so fully opaque, 8D8E98. And you'll see that this is kind of this light gray color, and again all of these colors came from the original Dribbble design just by using the color picker tool. So let's add in all the commas so that when we hit save, Dart will reformat our code for us. Now at this point if you look at our design, you'll see that we're not actually pulling in the font awesome icon, and this is because after every time you add a new package and you run packages.get, it's actually a good idea to stop the app and run it from scratch so that we do a cold restart and it can incorporate all the things we need, namely the font awesome icons. And now you can see the actual icon that we want onscreen but our column,that is the child of this reusableCard,it's kind of all over the place. But we can add a column property if you remember from before, something called mainAxisAlignment to actually center our column in the parent.

So if we hit save now you can see that our icon and our text will go right in the middle of the parent which is the reusable Card.

Now we know that in the final design, we're going to have two cards that essentially look pretty much the same.

They both have an icon and a bit of text and the styling are exactly the same.

So it makes sense then for us to actually reuse this part where we're creating the icon and the gap and the text widget so that we can extract it as a separate widget that we'll be able to reuse anywhere we wish.

So here's a challenge.

So we're going to create a custom icon content widget. And as a challenge, I want you to try and extract everything that is inside our cardC,hild which is going to be repeated across two cards, into a separate widget.

So try and remind yourself how we did it for the reusableCard and pause the video and try to complete this challenge.

All right.

So in order to do this, we're going to select the highest level of the widget that we want to extract.

So we know we have a column with some children but it's the column that we actually want to take out of the main tree and create a separate widget. As long as our cursor is selected on it,then we can head over to our Flutter outline and you can see here in the main widget tree, we've got the corresponding column selected as well.

And now all we need to do is right click on it and click on extract widget.

And now we get to give it a name and we can call that iconContent and we're going to refactor this into a separate widget.

Now as I mentioned before as of the latest version of Dart, we don't need the new keyword, but we now have a icon content widget.

And it's now extracted down into here.

So now the next challenge for you is we've extracted this widget but we need to be able to reuse it in two places.

So one we've already got which is here in the iconContent, but the second place is over here in the second card. Because the icon content describes a column that has a icon and a bit of text.

But of course the icon and text are actually different between these two cards.

So we need to be able to reuse that icon content widget to be able to display each of these no matter what icon they have. That means you're going to have to change a little bit of code that we currently have in our icon content widget.

The end goal is to be able to have that same icon content widget to be able to create both the male icon card and the female icon card by using that custom widget that we created just now.

So pause the video and try to complete this challenge. So in order to achieve this we're going to do exactly the same thing that we did in our custom reusableCard,that is create properties to be able to use in the widget that we build.

So over here we're going to first remove the default constructor that came when we use that extract functionality that Flutter has where we extracted our icon content, and instead we're going to create our own custom constructors so that we can pass over a value for the icon and also a custom value for the text.

So first of all, let's create our properties.

So again it's going to have the final keyword and we're going to see that whenever we create properties for our widgets.

And then we're going to have a data type,so in this case it's going to be icon data.

Now how do I know that it's going to be of icon data data type?

Well you can always check the icon widget and you can see that for the property called icon,it's expecting a icon data type.

So we've got our icon data and we're simply going to call it icon.

And now we can add our second property which is going to be a string.

And this is going to be the label that we're going to pass over which is going to go right here so that we'll be able to pass over a female when we're creating the female card, and male when we're creating the male card.

So now we're getting some errors because we have yet to initialize these final properties.

Remember that final variables can only have one value and that is at the point where they are created.

So when this icon content gets created, the part that gets triggered is of course our constructor.

So I'm going to create the constructor now. And the constructor is going to have some named properties and it's going to have a this.icon and also a this.label.

Now we could make it so that both of these are required but you can also just leave it as it is.

So now that we've created our constructor, we've created our properties,

It's time to actually specify where these properties are going to go in our main build method.

So the first thing that's going to be custom is of course our icon.

So it's no longer going to be hard coded as the mars icon.

And we're going to change also the text to the label property.

So whatever is passed in through this property is going to go right here.

That's going to be male and female.

And so now when we go into our BMI calculator, the main widget tree, we can update our icon content to include a icon which is going to be a font AwesomeIcon.mars.

And then there's also going to be a label which is simply going to be the word 'MALE'.

And that's why this expects a string.

So let's hit save and now let's go ahead and also do the same for our second card here.

So our second reusableCard is also going to have a cardChild, and the child is going to be a widget that's our icon content widget.

And it's also going to have an icon but this time, the icon is going to be the venus symbol, venus for women and the label is going to read female. Cool.

So now if we hit save and we head over to our app, then you'll see that we have both cards being created and all of the styling is coming from our IconContent.

So the size of the icon, the size of the gap between the icon and the text, all of that we didn't have to create again.

Instead we're relying on our custom widget IconContent here. And we're just passing in some different values for the parts which need to change, such as the actual icon and the actual label.

So you can see that we're kind of starting to create our own widgets rather than just relying on the out-of-the-box Flutter widget. And we're able to use these properties and our constructor to be able to specify some custom content for each of the widgets that we create.

At this point, you can see our custom widgets are starting to take up a lot of space in our main input\_page.dart file.

Now in this case it's actually probably quite a good idea to separate out our IconContent and our reusable Card in two separate Dart files.

And this is because we might want to use this in other files as well, maybe in other pages or screens in our app.

So it doesn't make sense to have it all crammed into one file.

So let's go ahead into our lib and create a new Dart file. And the first one I'm going to call icon\_content and then we're just going to click yes to add it to Git. And I'm going to also create a new Dart file that's going to hold our reusableCard, reusable\_card and it will create our Dart file as well.

So now let's go ahead and take everything that's related to our IconContent and cut it out of the input\_page.dart file and instead paste it in here.

And of course whenever we're using our material widgets, we always have to import the material package.

And we're also going to transplant our reusableCard over to our reusable\_car.dart but not before we import our material.dart package.

And now we can paste in our reusableCard.

Now if we take a look at our IconContent, you can see that we have a lot of things are pretty much hardcoded right? Like the size of the icon or the size of the gap between the icon and the text or things such as the style of the text.

Now it often makes sense to simply just take these parts out and create it as a constant at the top of the file.

And this way if we think that we want to change the labelTextStyle, then we have to see it right at the top here and we can update and edit it as we need to.

And inside here, we simply refer to labelTextStyle. And that way if we have lots of labels or needing that particular style, then they can all refer to this constant.

So the final thing to fix in here is that we have a lot of errors because it no longer knows what reusable Card is or what IconContent is because those widgets were removed from this file.

So in order to tell our file about those other widgets, we have to import the files that we just created.

So icon\_content.dart and also reusable\_card.dart

So now everything should be re-factored into separate files and it'll be much easier to work on each of these components separately if we need to.

So just as the native Flutter widget will have their own file,so for example if we hold down COMMAND or CONTROL on Windows and click on our text, and you can see we've got our text.dart file which contains all the code for a text widget.

Similarly we have our own custom widgets in their own files. So we can reuse it across our project.

All right.

So if you find the import getting a bit too much and taking up too much space, it's not a bad idea to simply just collapse it.

We never really need to refer to the imports when we're creating our code. And now our main input page has a widget tree that is mostly full of layout and you can more easily see at a glance what it does rather than having the tree completely exploded inside here and a million lines long,we have something that's quite succinct and it's quite easy to figure out what's going on at a glance.

In the next lesson,we're going to try and change the color of our card when we click on it, because at the moment nothing really happens.

It doesn't seem like as if I could select either of these.

So we're going to solve that problem in the next lesson.

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