All right. So as I said, the first thing that we're going to do is build a scorekeeper to keep track of　how many questions the user it got right and how many they got wrong.

Now in order to do this, we're going to place our code at the end of our column. So our column has a number　of children.

It's got a couple of expanded widgets, one which is for the question and two for the answer buttons.

Now right at the bottom here you'll see a To do. And this is what we've put into these starting file　to tell you this is where your code for your scorekeeper should go. To dos are really helpful because　when you're working on a large project, the first thing you probably want to do is to break it down into　smaller more manageable bite-sized chunks. And then you're going to tackle each of those one by one until　your final project gets built.

So it's really easy to break down a large project into a series of to dos. And in Android Studio, in order　to create a to do, it's as simple as simply writing a comment.

So our two forward slashes, and then we write the word to do, and then we add a colon and afterwards we　tell our future self or other colleagues that we're collaborating with, what to do.

So once you've created your to do, you can find them inside the to do tab down here.

So if you click on that, it will show you all of the to dos in your project.

And at the moment, there's only one inside our Quizzler project and I'll show you how to find it.

It's inside the lib folder, inside the main.dart file. But a far easier way is to simply just double click on the to do and it will take your cursor all the way to the place where it needs to happen.

First things first, let's delete this to do because we're going to do it right now. In future challenges,we'll be setting the challenges for you,each of the steps in the to dos.

So now that you understand how it works, you'll be able to use it to help you figure out what are all the steps you need to complete.

All right.

So now that we've deleted the to do, we are perfectly positioned to create our scorekeeper. And our scorekeeper is simply just going to be a row widget that's going to go into our column. And this row is going to have a bunch of children which are icon widgets.

So we're going to create an icon widget, and we're going to say that the icon that's gonna be displayed comes from the material icons. And the particular one that we're gonna show is something called check.

And you can see there's a little preview over here.

And when you hit ENTER, you'll see it show up over here in the gutter as well.

And this is simply just a tick.

We can change the color of our icon to a green color so that we can further make sure that the user knows that when they see a tick, it means they've got the last question right.

But we can also add a icon for when they get something wrong.

So we'll add a icon that is the one called close.

And this is a nice little cross for us, to tell the user that they've got the last question wrong.

And this one is going to be red of course.

So now that we've got a green tick and a red cross, if we hit save and check out our project at the moment,then we see our little row show up at the bottom.

So this is going to be our scorekeeper.

And as the user gets things right or as they get things wrong, we're going to add icons down here to keep track of their score.

So this row is going to be building up by adding new icons like this, until the point where the user is done with all the questions, and then we will clear this line out and delete all of the icon widgets.

Now we know that rows and columns can contain multiple widgets.

So in our column for example, we've currently got three expanded widgets and a row widget. And in this row, we've got five icon widgets. But other ones such as a safe area or a padding widget or a center widget,they can only have one child.

The difference between them is that the column widget can accept children in the form of a list.

Now a list is simply a way of grouping data into collections.

And this is a common thing that you can do across different programming languages.

So if you come from a different programming language, you might know list as an array. But in both cases,it's a way for us to be able to group related pieces of data into a single list.

And later on, we can manipulate the list or tap into its contents.

Now we're going to go ahead and create a empty list, and we're going to do that right here. In order to create a list in Dart, we're going to write the keyword list.

And this says that whatever variable that we're creating afterwards is going to be of type list, and then we're going to give it a name, so we'll call this scoreKeeper, and it's gonna be set to equal a empty list.

Now when you're working with collections, you'll often see these square brackets.

And in this case what these empty square brackets represent is just an empty list.

There's currently no items inside the list. But we can also add items to the list when we create it in the same way that we did it for all rows and columns.

So when we said that this row is going to accept a bunch of children widgets, we had our opening square bracket and our closing square bracket and in between are all the items in this particular list.

Now we can do that for our scoreKeeper as well.

I'm going to transplant some of these icons over, and I'm going to simply cut it and paste it in here.

So now we have a list that's called scoreKeeper and it contains five icons.

And later on if we decide that we want to use the list, then we can simply refer to it by its name.

So I'm going to delete everything on the right hand side of this children property.

I'm going to instead refer to my list I've created up here called scorekeeper like so. Now in this case, because scoreKeeper has a data type of list and the children property for the row requires a list,then these two sides left hand side and the right hand side, are compatible which is why this works.

Now notice that in this case, it's actually expecting a list of widgets whereas over here, our storekeeper is a list of data that have a dynamic data type.

Now as we mentioned earlier, when you're using dynamic data types and a lot of things are happening, you can very easily get into hairy situations.

So the best thing is to actually explicitly state what type of list it is that we're creating.

And by that we mean what kind of data can actually go into the list.

So for example, if I was to create a list of strings, then I could call it my strings.

And when I create it, it would contain strings like a and b and c.

Now in this case, the way that I could specify that this particular list called myStrings can only accept strings inside the list,I can do that by simply adding an angle bracket after the word list, and specify the data type that goes into the list.

So in this case, it will be like this.

Well now I can, for example, create my list and start putting numbers in here. Or rather integers, because it says that int can't be assigned to the list of type String.

So this makes our list a lot safer so that we don't accidentally put in the wrong data type.

Now up here, it's the same thing.

If we want to specify that we only want a list that contains widgets, then we can simply add the word widget inside some angle brackets, right after our list.

Now in this case, because we know that everything that's inside this list are icons, we can also say that this is a list of icons. And this still works down here because icons are in fact widgets.

So if this particular row can only take a list that have widgets, it is still compatible.

Now if I instead had a list of strings, which are definitely not widgets, then down here I will get an incompatibility error. Because a list of strings can't be assigned to a list of widgets, but a list of icons can.

So let's change that back. So now when I hit save and I check out my app, you can see that pretty much nothing has changed. My row down here is created using my list called scoreKeeper rather than directly creating it inside my row. The benefit of taking out my icons and putting it into a separate list means that I can now add icons into my list of icons on the fly.

So for example, I can say that when the user presses on the True button, then inside the onPressed, I'm going to get it to add an icon to my list of icons.

First of all I have to tap into my scoreKeeper list and then I'm going to use something called .add, which adds a value inside the parentheses into the end of the list. And the length of the list will grow by 1.

So we're going to say scorekeeper.add,and the thing that we want to add is a new icon.

Remember that scorekeeper can only contain icons.

So we're going to create a new icon, and this icon is going to be a check sign. And it's going to have a color of green.

So now that I've created my icon and I've added it to my scoreKeeper, all I have to do is to put it inside a set state so that it will trigger a rebuild of my widget tree.

And that new icon will show up inside my scorekeeper row.

So let's go ahead and hit save to update our code. And now when I click on the true button, you can see a little green checkmark will get added to my row. We've been able to do this because of the power of Dart lists.

Now if this concept of lists is new to you, then I recommend heading over to the next lesson where we're going to dive deep into Dart lists and we're going to learn more about how it works.

Now if you're already familiar with how lists or arrays work and this concept isn't new to you, then feel free to skip the next lesson and head over to the lesson afterwards where we continue building out our app.