Now in this lesson, I want to talk to you about one of the most fundamental widgets that we're going　to be using all the time, to lay out our apps.

And that is a container widget.

Now a container widget, for those you guys who have done a little bit of Web Development or Web Design,

is very similar to a Div.

It's essentially just a layout box.

You can position it and move it around on the screen, and then you can give it a child to display such as an image or a piece of text.

Now whenever you're using a new widget from Flutter, my top tip is to always start with the documentation.

As I said previously, Flutter's documentation is really good. And all you have to do to find it, is head over to flutter.dev and we're going to go over to the widget catalog. And again, you'll find this URL to get here directly, in the course resources.

Now once you're here, you can see that the team has organized all of the widgets based on what they do.

And we've already seen some of the images and assets widgets, but now we're going to be looking at the layout section. And these other widgets that allows you to arrange other widgets. So how to position things on screen. And under the layout widget,you can see that they are split by single-child widgets and multi-child widgets. A widget that can contain a single widget, say a piece of text or an image, or lots of different widgets, such as five images or an image and a piece of text.

And in this module, we're going to be exploring both single child, and multi-channel widgets to learn how to lay out your screen, exactly the way that you want it to look.

The most commonly use single child widget is probably going to be a container.

And if we click on it, we get taken to the container class. And this is a widget that allows us to position other widgets and size other widgets.

And if you read through this, you'll learn a lot about exactly how it works.

And one of the things that it tells you is that, containers with no children try to be as big as possible.

And this is exactly what we're seeing over here.

We have a container that is currently trying to be as big as possible.

How do we know that?

Well let's give it a color.

Let's see it for ourselves.

So let's go ahead and change this back to teal, and instantly, we have our background using hot reload.

And now inside the container,so inside these round brackets, I'm gonna hit ENTER and I'm going to change its color. And here we get to specify a color for our container.

Let's change our container to have a color of white.

So now when I hit save and hot reload, you can see that my entire screen changes to white, even though we know that the background color is still teal.

And the reason for this is because our container is taking up all of the space that's available, exactly as described here.

Containers with no children try to be as big as possible. Our container currently doesn't have anything inside the child property.

So that's why it tries to be as big as possible.

But what if we did give it a child.

What if we gave it a child widget that's a piece of text, say that just says 'Hello.'

Right.

Let's hit save, and immediately our container shrinks to the size of the child widget. And we could have known this by reading the documentation. Containers with children size themselves to their children's size. And our child in this case, is the piece of text and its size is determined by the font size of that piece of text.

Now at the moment, you can see that all piece of text is kind of hidden away a bit by the bezel of our iPhone screen, and on Android depending on which device you're simulating it on, it might also have a notch like this.

So how do we keep our content that we put on screen, safe from all of these bordering elements like the top bar where we've got our time and our signal and our notch? Well Flutter actually comes with a really simple widget that does exactly this.

It's called a safe area. And anything that you put inside the safe area, will be inside the parts which don't have a notch or an interactive area like this part on the iPhone here.

In order to embed our container inside a safe area, we can hold down OPTION or ALT and hit ENTER, to pull up our intention actions here, our little menu. And I'm going to wrap our container, which is what's currently selected, inside a new widget. And all that does, is it puts our previous container inside a new widget,and it sets it as the child of that widget.

So now, we get to change this widget and I'm gonna call it a Safe Area, with a capital S and a capital A.

Now as soon as I hit save, you'll see my container moves in to the safe area and it now only takes up the space where it's visible to the user.

Whenever you're creating something for the user interface that you don't want to bleed all the way to the edges but instead you want it to be fully visible to the user,then you're gonna be using your safe area.

What else can you do with your container?

Well we can also change its size, such as its width and its height.

So if we just go ahead and add a new property called height, and let's set it to maybe a 100 pixels,

100 points. And let's change the width also to a 100.

So now we should have a 100 by 100, so a square container, as soon as I hit save.

And there you go.

We've got our square container, with a single child that is the piece of text that says 'Hello.' Now what if I wanted my container to be a little bit offset from the edges of the screen.

What if I wanted to set a margin for my container?

Well, I can use the margin property. And to specify the margin property, we have to use something called edge insets.

And you can see there's different types of insetting an edge. You can use the all,which means that whatever you enter here, let's say 20 pixels, is going to be set for all four sides of your container.

So there's now a little margin of 20, for the left, the top, the right and the bottom.

Now what if we didn't want that.

What if we wanted to only set it for the top and the bottom?

Well, then you can use symmetric.

So now you can say that for the vertical axes, I want it to have a 50, but for the horizontal axis, I only want it to have 10 for the margin.

So now if we hit save, you can see that it's offset from the top by 50 and from the left and right it's only 10.

And you can always view this easily by going to the Flutter Inspector and clicking on this button here.

And now, you can see the margin for our container at the top and bottom being 50, and the left and right being 10. And you can also see the safe area which is this box over here as well. Now instead of symmetric where you provide the same value for the top, bottom and the same value for left and right, you can also give it different values for each side. So you can either say, 'from LTRB'

So this is left, top, right and bottom. Left,let's say it's 30, top is 10,right is 50, bottom is 20. Every single side

I've given a different margin. And now, when I hit save, you can see how our box changes right.

You can see that the left is 30, the top is 10, the right is 50 and the bottom is 20.

So this box here, is what our margins are doing to our screen. But if you didn't want to have four values specified, you can use the 'only'. So saying 'only' gonna set a margin for the left, which is 30. And now everything else is left as zero and only the left gets that margin.

So there's all sorts of ways that you can do this. Now, what if you wanted to have the child of the container to be a little bit more offset from the edge of the container.

In this case, you wouldn't be using the margin and you would instead be using a padding.

So if you've ever worked with CSS or Web Design, then this will all feel very intuitive to you.

And remembering that the Flutter team actually came from a background of Web Development, this is one of the reasons why it might be so familiar and so intuitive to work with.

So for the padding, it also expects a edge inset.

So let's go ahead and provide one.

And this allows us to set basically different values for the padding, for all four edges of whatever it is that we're setting it to, which in this case is our container.

Let's add a padding for all sides, that's 20.

And when I hit save, you'll see that I now have a little box inside my container, that's limiting the child to this area right here.

And if I go ahead and remove all of these guidelines, then you can see that this is what I end up with.

I've got a container that's pushed slightly away from the edges of the screen, and I've got its content pushed away from the edges of the container.

So margin is for the outside of your widget, and padding is for the inside of your widget.

And remember that a container can only have one child.

Right.

So for example, I can't have a piece of text and maybe say an image. That won't work. Because remember the widget itself, can only have a single child.

So we're only laying out a single child. In the next lesson, we're going to look at some of the ways for laying out multiple children namely, using widgets such as columns and rows. So all of that and more, I'll see on the next lesson.