All right.

So now that we've seen classes and objects in action and we've used them a little bit, it's time to do a deep dive and understand exactly how classes and objects work in Dart.

So firstly a class is simply just a blueprint and it's used to create all of the things that we're going to be showing in our app and the user is going to be interacting with. So essentially, everything that makes up our app is going to be created using a blueprint that's called a class.

Now what exactly is an object then?

Well when we take that blueprint and we take it to the factory and the actual car, in this instance, gets built, well that's the object. That's the actual thing that's going to be doing the driving, or in our case of an app, it'll be the actual button that can receive taps and the images that actually change.

Those are the actual objects. But because we can't see the objects until the app is running, we have to plan out how they will function and how they will work by creating these classes.

A class has two really important things. One are its properties.

So if we're building out a car, then that will be its color or the number of seats it has or the number wheels it has or the number of doors it has or all sorts of things that are things to do with the characteristics of the object. And in Dart, these things that we've been referring to as variables when they're inside a class,they're now called properties or fields.

Now the other important part of a class that we need to define are its methods.

So the things that it can do. The actions that the object that will eventually get created, will be able to perform.

In the case of our car object, that might be to drive, or to brake, or to turn around.

And these methods are actions that the object will be able to do that we're going to define in the blueprint even before the object gets built.

So for example, you might see a class defined like this.

Firstly you have that class keyword which shows that everything else that's coming afterwards is going to be a blueprint.

And then you have the name of the class.

So in this case we're building a car blueprint. And by convention in Dart as with many other languages,when you're creating a variable say here we've got numberOfDoors, or when you're creating a function,say here we've got drive,those are all camel case starting with a lowercase word.

But when you're creating a class to show that it's different, we always start-off with a capital letter.

So all the words in your class names are capitalized.

Now here's our Car class and inside these curly braces are the definitions for this car class.

So this is the blueprint of exactly how our car will look and also how it will behave.

And between the looks and the behavior, we end up with a class.

So in this case we have just a single property, which is called numberOfDoors.

And this defines a characteristic of the car namely, how many doors it has.

This car has five.

This car also has a function called drive,and this defines a little bit of functionality for the car.

So namely this car, when we call this thing called drive, it's going to start turning the wheels for example.

Now notice how inside a class, our variables are no longer called variables and they're called properties or fields.

Similarly, a function when it's defined inside the curly braces of a class, it's now called a method and not a function anymore.

So if you're new to programming there's a lot of new words and it's a good idea to build your own glossary of words because it's just like learning a new language.

When people refer to a method, you can think of them as just a function that's associated with the class.

And when people are talking about a property, it's simply a variable that's associated with the class.

And once you hear it often enough, then it'll start becoming second nature.

Now once we have our object, it's the object that actually performed those pre-planned actions in the class such as moving, right?

Or it actually has those characteristics which we defined in our class such as its color property or the numberOfDoors property. The two paths of creating a object are first, we have to create the class and this is gonna be the blueprint that's going to define everything about the eventual object that will get created.

So we use the class keyword, we give the class a capitalized name and we define variables in it which are properties for that class.

And we also give it functions which become the methods that the class can do.

And when we actually create the object from this class, then our code looks something like this.

We would define the type, which is in this case, the name of the class and then we would give our object a name.

And finally we would set it to equal a new version of that class. When we have the word car and these empty parentheses,it's basically creating a new object from this blueprint.

It's like sending that blueprint to a factory and out comes our actual car object.

Now this is all very abstract and we're having to imagine a lot of it in our head.

So let's put it down onto code and head over to our trusty DartPad to be up to see this in action and understand a little bit better.

So I've got my empty main method up here, and I'm going to create a new class right below it.

I'm going to call that class Human.

So remember it should be capitalized.

And this class of humans are going to have some properties, so some characteristics.

So what kind of characteristics might a human have?

Well they might have a height right?

We could create a height property and maybe it'll have an age property. And when we create a new class,we can either create these properties as empty or null such as here, or we can say that by default when we create a new human, it should always have a height.

Let's say that when a default human gets created, it starts out being, I don't know ,maybe 15 centimetres in height, and maybe it will have a default age of zero.

So now if I go into my main method and I create a new object from this human, let's say that we create a new human,so that's gonna be the type of this object. And our actual human, let's call her Jenny, is going to be created using our human class. And we do that of course by writing human and then we have a set of empty parentheses.

And this builds out our new human.

So human is the blueprint for all humans.

They all should have a height and they all should have an age.

And we have some default values for how humans should be when they first get built.

And Jenny is our actual object who's built from this blueprint. And we're doing things in our program,we're actually interacting with Jenny rather than the sort of abstract blueprint kind of for human. So we might say some things such as, let's print Jenny's height. So we have our print statement and then we're going to tap into our Jenny object and then we're gonna use the dot notation to tap into its property namely the height property.

So we'll say jenny.height and let's print that into the console. And you can see that because we've given all humans a default starting height of 15,Jenny also has a height of 15.

Now a little bit later on, I could maybe change that.

Maybe Jenny grows a bit and lets say that her height is now 20cm.

So I can change all of these properties and I can tap into them simply by using that dot notation. By saying that this is the object that I'm concerned about and it finds the object which we know gets created from the human class, and by default, all of the properties of a human start out with these values. Then I'm saying get me Jenny's height. And that's what that dot says.

It's just saying, 'this is the property associated with Jenny that I want to know about.'

Or later on when our human get some functions associated with it, you'll see that the way that we tap into it is still with the dot notation.

So when you see something.something, it usually means that this is an object and we're tapping into one of its properties or methods.

Now at the moment, every single new human that we create starts out with a height of 15cm.

Now that might not be true right?

So let's say that we created another human and this is James.

And he's also created from the human class, which is what this part means. We're going to the human factory and we're building a new human.

And if I was to print his height, you'll see that it's exactly the same.

He's also 15 centimetres tall. But that's probably not really true right?

Why should it be that when a new human gets created, they all have the same default height?

What if we wanted to initialize each object with a different value for some of these properties?

How would we do that?

Well we already saw that in the last lesson that we can use something called a constructor for our class.

And to create a constructor,we would write the name of the class, so it's human, and then we would create a set of empty parentheses.

And just as when we create a function we provide some inputs into these parentheses, when we create a constructor for a class we can provide some inputs to our class.

So we could say that maybe the starting height property which is going to be a double, because our height property is a double.

We'll give it a name of startingHeight.

And now inside the curly braces, we can assign these startingHeight to the height, and height can start out without a value. It'll start out being null.

But when we create our new human, we have to provide a starting height and then we can say that the humans

height property will be set to whatever the starting height is that we provided.

So now we get some errors here because it needs a starting height and we haven't given it one.

So let's add that in. So we could say that Jenny is a human who starts out being 15 centimeters but James starts out being a human who is, maybe he's a bit taller. Maybe he's 20 centimeters.

So now when we click run, you can see that Jenny's height starts out 15cm and James's height starts out at 20cm.

And our code now makes a little bit more sense because why should a human have a default height right?

They they don't all get created with the same height. And just as we could make the input or the parameter a named parameter, in our functions we can do the same in our class constructor. So we can add a set of curly braces around these inputs to say that when we use it when we create our new human object, we have to specify that name.

So startingHeight and then a colon.

So this way it's a bit easier on the eye and you can see that I'm creating a new human and I'm providing a starting height for that human. As I said earlier, our classes can have properties and it can also have methods. Our methods are simply functions that are defined inside a class.

If we were to create a method let's say, talk to get a little human to say something, we can tell it what to say.

So we'll create a string input called whatToSay, and then inside the talk function or in this case the talk method because we're talking about a function associated with the class, we can say that we'll just print the string that it's told to say.

So that means we can tell our new humans, say for example, let's get rid of James. Let's just say Jenny

.talk and the thing that you should say is ,Why is this sky blue?'

Now when we run our code, we print Jenny's height and we get Jenny to talk. And you can see that by calling this method talk, then this object actually performs that action and it prints out the thing that we want it to say into the console.

This is now a method that's associated with the human class.

And just as we use the dot notation to access the property height, we're also using the dot notation to trigger the talk method.

So when we're working with objects you see the dot all over the place and it just refers to saying, 'this object do this or this object get me it's property.'

So we now have a pretty simple class. And this class is our blueprint for how a human object should behave and what characteristics it should have.

So all of the things that a human object, once it gets created, should be able to do and what it should be. You can of course make this as complex as you need it to be. But in essence, we're simply creating a plan for how this object should behave or how it should appear. And we use it by creating it as a new object and we can create as many of these as we need to and get them to interact with each other or get them to do specific tasks.

And so this is how we would create our classes and objects.

But why exactly do we need these classes and objects? To find out about that,

I'll see you on the next lesson.