Now in the last lesson we learn all about Dart enums. And our enums are making our code a lot easier to read and it's starting to make a lot more sense.

But there's still so much code for achieving some functionality that's actually really simple which is toggling a card's color.

Why do we need so much code? In order to reduce the amount of code but still have the same logic and have the same behavior,we need to learn about something else that Dart gives us, which is the ternary operator.

Now previously we saw that we could use the IF, ELSE IF, ELSE statements to be able to perform a different behavior depending on what the condition was.

And we talked about it in terms of a rail track right? Deciding which way to go depending on which conditions were currently in play.

Now the Dart ternary operator is actually just a bit of what we call syntactic sugar to make something that's actually very long and very wordy, say an if condition is true then do this else do this.

And there's a lot of brackets, there's a lot of keywords and there's a lot of lines of code to achieve this. But if we used a Dart ternary operator instead, then it simplifies that into a single line where we check the condition, if that condition is true then we carry out this.

And if the condition was false then we carry out this. And all that we need here is a question mark and a colon. If that seems a little bit magical don't worry because we're going to head over to our trusty DartPad to flesh it out and see how it works.

So let's say that in our main function we have a new variable and it's of type boolean, so it can be true or false.

And let's give it a name.

Let's say that we're trying to track to see if jackBauerIsAwesome, which obviously is true.

So if anybody doesn't watch the show 24 then just ignore me.

Just pretend that this is a normal boolean variable. So you can just change this to a different variable if you want,say durianIsAwesome.

It doesn't really matter.

It's simply just a variable that can be true or false.

Now at some point in our program maybe this gets changed to false or maybe it gets changed true,so we need to check for it using an IF statement,right?

We might say something like if jackBauerIsAwesome then if that's equal to true, then maybe we would want to print

'That's great.'

Or else if that's not true maybe we want to print

'Oh no.'

And let me just make sure I've got all the quotation marks where it's needed and escape that quotation mark there.

So now if we run our code, it will print that's great because jackBauerIsAwesome is equal to true.

And we check to see if that is indeed the case.

And so it falls into these brackets and it prints 'That's great.'

So this is classic IF ELSE statement behavior. Now we can cut down all of these lines of code into a single line just by using a ternary operator. And in order to do that the first part that comes is the condition that we're checking for.

So we're saying jackBauerIsAwesome is equal to so double equals remember checks to see if the left hand side is equal to the right hand side.

And if this is true, then what we'll do is print that's great.

And we add a question mark to say that this is what we're checking for.

Well if that is true then we're going to print that is great.

But if that is false, then we're going to add a colon to say this is what we should do if it was false.

So we're going to print oh no. This line of code does exactly the same thing as all of these lines of code.

And I can comment that out and hit run to just confirm to you that it's exactly what happens.

And if I change this to false, then we should now be triggering this part and it will print 'Oh no'. Pretty cool right?

Cutting down on a lot of code.

Now the first time when you see this in the wild, it can be a little bit confusing because it's not quite as laid out as all of this. Now,we can also cut down this line even further because the computer will look for all the parts before the question mark to see if it's equal to true.

So we don't actually have to check if it's equal to true.

It's the same as in here in the IF statement.

We don't actually need that equality check. As long as this has a value of true, then it will trigger this part, but if it has a value of false it will trigger this part. Now you always want to avoid playing code golf where the only goal is to cut down on the number of lines of code and to make things as short in the number of characters as physically possible.

This is an actual game, code golf, but that's not really the goal.

Most cases you want to make sure that your code is succinct but it's still expressive.

It still shows what's actually going on.

So whenever you're choosing between whether if I want to just cut down the lines of code or whether if I can cut it down and make it clearer and easier to read and less confusing, then that's always what you want to choose.

So we're not just doing it to cut down on the number of lines of code.

Now what we can also do is use the same comparatives that we can use in our IF statement using our ternary operator.

So let's say that we create integer called myAge and let's say it's 12.

Don't worry I'm not 12.

It's fine. But if we wanted to say create a boolean that is called canBuyAlcohol, and we want to assign it a value of true or false, depending on the value of my age then we can also use a ternary operator here on the right hand side.

So we could say something like if myAge is greater than 21 ?

So we're going to check to see if this condition is true.

Well if that is the case, then we're going to set canBuyAcohol to true.

But if that is not the case, then we're gonna set it to false. This is something that the IF and ELSE statements can't do. We can't put down multiple lines of code on the right hand side of an equal sign.

But because a ternary operator is simply just a one line expression, then we can in this case.

So if I go ahead and print canBuyAlcohol, then you can see that while my age is equal to 12 we will get false and if I change this to a little bit larger, then you can see that we get true.

So we're basically assigning this value to this variable depending on this condition.

And we're doing this all by using our ternary operator.

So in most cases, this is what you will see happen.

So this is super useful when we need to check something and depending on that check, we give our variables a different value to work on. Back to our code,it's again challenge time. And this challenge is a little bit harder but essentially I want you to apply what you've just learned about the ternary operator and I want you to get rid of this function. Instead simply use a ternary operator to check in place right here where we have our color and to check if the condition is that the maleCard is selected then we should give it an active color but if the female card was selected, then the color should be an inactive color and it should be the opposite over here.

So we should have exactly the same functionality as what we've got right now but without the need for this function. And you might need to add some custom variables you might need to change the set state a little bit and you definitely need to use a ternary operator here to check for which value color should be.

So it's a little bit of a harder challenge but I think you can do it. Pause a video and give it a go.

All right.

So first things first.

I said we're going to delete this function.

I'm just going to comment it out for now. And we won't really need this, but instead we're going to create a new variable that's going to hold a gender and it's going to be called selectedGender.

And it's going to start out being null, because no gender is selected to begin with.

But when the male card is pressed or when the female card is pressed, we're going to change the value of the selected gender. When this card is selected, the select agenda is going to be equal to gender.male.

And when this card is selected, the selected gender is going to be gender.female. And we're able to write this expressive really clear code all because of our enum that we have.

And that's how we're able to have this property that's holding a gender value rather than just an integer or something arbitrary.

So now that we're able to set which gender was selected and keep track of that value using our variable called selectGender, then we can now address how we change the color of the card.

So we know that we can use our ternary operator as a way of evaluating a condition and assigning a value.

In this case for our male card,we could check to see if the selectedGender is equal to gender.male. Then in that case the color of our reusableCard should be the activeCardColor.

But if this was not the case i.e. if selectedGender equals male is false, then we should change it to the inactiveCardColor. And in our reusableCard for our female card,the opposite is true.

So we're going to check to see if the selectedGender is equal to female. Then in this case, we're going to change this reusable card to have a color of the activeCardColor but otherwise i.e. if the male gender was selected and this was false, then we would change it to the inactiveCardColor.

Now this also has the benefit when our app first starts out,gender has no value.

So both of these statements will evaluate to false. selectedGender is not gonna be equal to male and it's also not going to be equal to female.

So both cards will start out with the inactiveCardColor, which is exactly what we want now.

I'm going to go ahead and delete all of this code and now it's replaced by just these few lines of code.

Super simple, super easy.

And we're able to do this all because we're able to evaluate for an expression and assign this color property a different value depending on that condition all in the same line because we're able to use a ternary operator now instead of having to rely on IF and ELSE. Let's go ahead and test it out.

So I'm going to click on hot restart so that we reset the state of our app to the very beginning.

So that means that this gender, selectedGender is going to start out being null, which means both our cards are going to be inactive. But as soon as I select one of the cards, it becomes the active color.

And if I select the opposite card, then it'll will deactivate that card.

So we have exactly the same functionality and it's created using just five lines of code.

