All right.

Let's talk quickly about Dart maps and how they work and how we can create them and what we can use them for. When we're talking about Dart maps it kind of refers to not like a map of the world, but more like how something can map onto something else.

So for example if I type this last June longitude into Google maps, it will map onto a specific location.

And even better than that, you should think of maps as almost like a phone book right?

Let's say that I have three friends.

I hope you have more friends than I do.

Amy, James and Tim and I keep their phone numbers in a book because I'm old school like that.

Now a phonebook is kind of like a very basic sort of map.

It's able to associate two pieces of data together.

So usually you would have a key, in this case it'd would be the names of my friends,and then you would have some values and these would be the values that are associated with the keys.

This is how maps are. They're kind of like a collection type just like lists which we saw earlier.

But unlike lists, maps are unordered. So they don't go from zero to one to two because it doesn't make sense to order your names in your phone book unless you're super OCD and you really want them to be alphabetized and you would rewrite your phonebook every single time you had a friend that had a name beginning with A. But essentially all that maps are is we have a key and we have a value and the key will map onto a value and we can pull out the value if we know the key. In code,what this would look like is something like this.

So they would be contained inside a set of curly braces.

We have a key and then after a colon we have the value that's associated with that key. And all of these entries are separated by a comma.

So this is how we would create it and this is how we would use it. If we wanted to create a map, we can specify the data type as a map data type, and then we can have a set of angle brackets and just like when we had lists where we put in the data type all of the values that are in the list,in this case we can specify the data type of the keys and the data type of the values so that when we create our map or when we try to add values to our map our compiler can check to make sure that the key fits the initial key type and the value fits the value type.

And if we want to use our map we simply refer to the name of the map,use some square brackets and provide a key. And this will end up giving us the value that's associated with that key.

So let's create a new map from scratch.

I'm going to start out by providing the data type which is going to be a map data type. And I could just leave it at that and get Dart to figure out what are the data types of the entries in the map.

But if I wanted to help it along and also help other programmers who I'm working with to be able to see at a glance what is contained inside this particular map, then I could add some angle brackets. And inside, I could provide the data types of my key first which in my case is going to be a string, the names of my friends. And then the value is going to be an integer,the phone numbers of those people.

And so this is how that full data type looks like. I'm creating a collection of type map and it has a key that will be string, and it'll have associated values which will be an integer. And then I get to give it a name.

So let's call it I did a phonebook right?

And it's gonna be set to equal a new map.

So I'm going to open out with some curly braces and inside here is where I will specify my keys and values.

Now in this case my keys have to be a string.

So let's say that we create a key called I don't know, Kyle, and then we would add a colon to say that this is the end of the key.

And now I'm going to write the value that's gonna be associated with Kyle, so let's say that he's at some random phone number. And then I can separate each of the entries in my collection with a comma.

So I'm going to paste in the ones from our previous keynote just for time sake.

And now if we cap off the end of our map with a semicolon, we can reformat our map to make it look pretty.

So here we have a map of strings and integers,it's called phone book,it's contained inside a set of curly braces and it's got four entries.

So now what I can do is I can start using my map.

So for example if I wanted to print out the value that is in my phonebook for maybe let's say Amy right?

Then I could write phonebook, add a set of square brackets denoting that I'm trying to pull something out of a collection, and then I would provide the key which obviously has to match one of the keys in my map.

So let's try and get Amy's phone number printed into the console. So let's hit run and we can see that we get 123456789, which is the value that's associated with the key Amy.

Now if I tried to pull out a key that doesn't actually exist in the map, let's say try to get my own number out of this phonebook, then all we'll get is null. When it tries to pull out of value for a key that doesn't exist you will get the value null.

So it's easy to check against this if you wanted to see if a particular key existed. You could just see if phone book at a particular key is equal to null, do this.

Else do that.

Now we can also use this particular syntax,the square brackets, to add new values into our phonebook,that doesn't yet exist.

So let's say I write phonebook and I'm going to add a value for my own name now.

So I'm going to add phonebook for the key Angela is now going to be equal to,let's add some random numbers.

And now if I tried to again print phonebook for the key Angela, you can see that this now works and I no longer get null printed in here. So the cool thing about maps is that it's unordered.

It doesn't really matter at which point your entries go in because you don't need to pull them out in order, as in 0 is Kyle,1 is Amy. And instead you can actually get the values out of the map using something that's quite easy to read and easy to understand when you're looking at it at a glance.

This syntax is very expressive.

Now there's a number of methods that make it easy for you to use a map.

So for example you could write phonebook.length to see how many entries you currently have in the phonebook.

So that's four plus this one which is five.

Or you could write phonebook.keys to print out all of the keys that's in the phone book. And it will print out everything it's currently got. Or you can do the same for values as well and it'll print out all the values it has. So we're going to be seeing maps a lot more in the coming lessons.

And it's just a convenient way of putting items into a collection not in any particular orde,r but searchable by their keys.



