In the last lesson was the first time that we actually came across this thing called a stream. And you　can see the effects of using it where we were getting our method being called again　every time we got a new piece of data that went into our messages collection. And I mentioned that this　is now effectively like we are subscribed to this collection and whenever there's new data going in　there, we're getting notified in our app. So what exactly are these Dart streams that we're using?

Well let's start off with an analogy.

And knowing me, it's going to be a food analogy.

Now let's imagine that you're in a Japanese restaurant and you decide that you really fancy having some　sushi.

Now of course it takes time for the chef to prepare your food.

You have to cut up the fish and make the rice balls etc. So you're going to have to wait a little bit.

Now there's two ways of waiting on this right?

One is that you could say, 'Well here are all the things that I would like to eat.'

And you make your order and you get a receipt.

This is your order number.

Once it's ready you'll be able to collect it.

And this is effectively how futures work.

And we already saw that and went into that in a lot of detail in the Clima module.

But consider another scenario.

Instead of saying, 'I want all my order in one go Please give me everything I've requested' in one go.

What if instead you had a seat at the table?

So let's imagine you're in a conveyor belt sushi shop and the chef will make these dishes.

And as he makes them, they get sent down the conveyor belt and you sitting somewhere downstream of the　conveyor belt will be able to pick up these pieces of sushi as he's making it.

This is kind of more like a stream now, because again, there is an event which you're waiting for namely　the chef having to finish preparing the sushi. But he's sending those bits of sushi out as soon as he's　finished preparing with it.

So by subscribing to this stream or by sitting down at this conveyor belt sushi table, you're essentially　able to get the pieces of data as they come in and as they get added to the stream.

So this way there is still an event that you're waiting on depending on how long it takes for the chef　to make the sushi. But instead of getting everything in one go and waiting for that final order to take　home,　well here you kind of just getting it as it comes through.

So that's how our messages have been coming through. We're subscribing to a stream of messages and that　code lives inside the snapshots method.

When we have a look inside the definition for that method, you can see that they're creating a method　which returns a stream of query snapshots.

And as more data get added to this collection, then the data will come in through our messages stream　and we can listen to it to either print out the latest piece of data or just print out all of the data　as it is inside our messages collection.

Now another way of thinking about this is if you thought of this as maybe a 2 x 2 matrix right?

So let's say that we have things that are already available,　so things like your normal objects, your int, your doubles, your strings.

And then there are things which you need to wait for. And let's say we wanted the singular version of　those things.

Well a singular version of something that's already available might just be a piece of data that's a　string object right?

Like a variable or a constant.

But what if we had a singular object that we needed to wait for?

Well that might be a future String object.

So this is something that will be processed asynchronously.

It might take a while for us to grab it from the internet but once it arrives, then we can use it as　a normal string.

But meanwhile when we're will encoding it up and when we're planning for it, the data type is a future string.

But what about plural items or multiple items?

So the readily available multiple version of a string would be a list of strings,　right?

It would just be a list that contains many strings.

But what if we have to wait for this list of strings to come through?

What is the plural version of a future string?

Well that's a stream of strings, just like sushi that's being added to the conveyor belt that could be　added to the stream at any time.

And as long as we are subscribed to that stream, we'll get notified every time there's a new piece of　data that's been added to the pipeline or the stream. And if you haven't had enough of food related analogies,　there's this really awesome thing called Nagashi-somen which is this thing that people do in Japan in the　summer where they split a bamboo open and they run water down it like a little stream. And somebody will　be upstream putting usually noodles down the stream and they'll flow down in through the stream.

And people who have subscribed to the stream　well they can catch the noodle as it flows down and add it to their bowl. So they're effectively using　a stream and listening to the data that's coming through.

Now you don't know how regularly the data is going to come through right?

It could be every second,　it could be one second and then maybe five minutes later.

But as long as you're subscribed, you're gonna get notified of when data is coming through.

So similarly over here, we know that the rate at which users will type messages is kind of irregular,　right?

Maybe I'll message my mom today but then I won't message her like for a month.

Well it doesn't make sense to keep pulling the data by using something like get documents because then　you would have to maybe put it on a timer.

Maybe we would have to check are there messages now? Are their messages? Are their messages? And you'd probably　have to do that every second or every fraction of a second which is really really intensive.

But instead, if we had a stream, then we would basically just be notified as soon as there's new data　and then we can process the data　namely by adding it into our app. So I hope that clears up streams a little bit more or at least tells　you a little bit more about what's going on over here.

And once you're done messing around with the code, head over to the next lesson where we're going to　talk about how we can turn our stream of data into actual widgets that we can display on screen.