Asynchronous data, streams, and futures:

Think of a busy restaurant where the waiter takes your order and then goes to serve other tables while your food is being prepared. You don’t have to wait for your meal to be ready before the waiter can take other orders. This is exactly asynchronous data. Just like the waiter can serve multiple tables, asynchronous data allows a computer or application to handle several tasks at once.

Now let’s understand the technicalities:

- In asynchronous transmission, each character or byte of data is framed with a start bit and a stop bit. The start bit signals the beginning of the data transmission, while the stop bit indicates its end. For example, when sending an ASCII character, it might be represented as 10 bits: 1 start bit + 8 data bits + 1 stop bit.
- This is cool, but it increases the memory that needs to be transferred because of the extra start and stop bits. And for the same reason, it also increases the processing overhead.
- Data is sent one character at a time, as opposed to blocks or frames used in synchronous transmission. This means that each character is transmitted independently, allowing for irregular intervals between transmissions.
- Unlike synchronous transmission, where data is sent in fixed intervals dictated by a clock signal, asynchronous transmission allows for variable timing between characters. The receiver does not know when the next character will arrive until it detects the start bit.

And a stream is like a hose that delivers water continuously. You can fill up a bucket (process the data) as the water flows out. The data sent by the stream is essentially asynchronous data.

Let’s understand futures in dart:

Future in Dart represents a value that may not be immediately available but will be at some point in the future and hence used to handle asynchronous data in Flutter.

A **Future** object can be in one of two states:

- **Uncompleted**: The operation is still ongoing.
- **Completed**: The operation has finished, either successfully with a value or with an error.