**Summary of the Article on Async**

**What does Async mean?**

First let’s just define a general definition of asynchronous programming. Asynchronous programming is about running code concurrently on a single thread. Multithreading was about separating the task between multiple cores of the CPU so the task can be completed quickly. But this method is useful when you are doing CPU intensive work like calculating or something. For I/O processes like reading from a file or waiting for a response from a server, multithreading can be quite expensive because when the threads are not performing the task they are just consuming some memory. Asynchronous programming let us run multiple I/O computation on the same thread. When a thread is waiting for the response it is in idle state, so we can let this thread do other tasks and the response is triggered the thread can get back to work. The values which take part in asynchronous computation and are waiting for the result are known as awaitable. In Rust these awaitable are known as ‘futures’

**Async Function Does Not Start Immediately!**

An asynchronous function is started when you call ‘.await’ or launch a task using an executor. Until this happens, you just have a function that has not started.

**Different Library**

You need an external to library to do asynchronous programing in rust. In Rust you need an executor. The executor is what takes care or running the asynchronous functions/futures. The standard library does not come with the executor, so we need an external crate to get this functionality. The two prominent ones are ‘async-std’ and ‘tokio’.