**Promises and asynchronous functions**

**Promises**

When we write an asynchronous function which is going to retrieve some data, we need to wait from that retrieval to finish before we handle the data.

A **promise** is designed to help us with that. (Note: Callbacks have the same function, but when there are going to be many chained callbacks – i.e. "callback hell" – promises will help clean up the code.)

A promise asks for something to happen.
A promise has *two* callback functions, usually named resolve and reject.
If the promise succeeds, then resolve will be passed some data and will execute.
If the promise doesn't succeed, then reject will be passed an error message and will execute.

It also has two methods, when are called after resolve or reject has executed: .then() and .catch() The then() methods can be chained, and the catch() method is chained after all the then() calls. There is also a .finally() method which is after all of them --- this should remind you of the discussions we had about jQuery.

If the promise's request is successful (i.e. it is in the fullfilled state ) the .resolve() method will be called.
If the promise's request is not successful (i.e. it is rejected) the .reject() method will be called.
After either of those happens, you may, if you wish, continue processing with one or more .then() methods, then a .catch() and optionally a .finally().

At it's most basic, a promise looks like this:

At <https://www.geeksforgeeks.org/javascript-promises/> we get several examples.

There are also examples at <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise#examples>

And <https://javascript.plainenglish.io/what-exactly-is-a-promise-601c6f39a8ad>

Notice that

* A promise is an object which is in one of 3 states: pending, fullfilled, rejected. (Some authors add a 4th state: settled)
* After the 'pending' is finished, the promise moves to either resolved or rejected.

**Asynch functions**

These are even cleaner ways to write functions which are based on promises. The MDN <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function> has a very clear discussion.

**References**:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise>

<https://javascript.plainenglish.io/what-exactly-is-a-promise-601c6f39a8ad>

<https://www.digitalocean.com/community/tutorials/understanding-javascript-promises>

<https://www.javascripttutorial.net/es6/javascript-promises/>

<https://www.javascripttutorial.net/es6/javascript-promises/>

<https://www.geeksforgeeks.org/javascript-promises/>

<https://www.freecodecamp.org/news/javascript-es6-promises-for-beginners-resolve-reject-and-chaining-explained/>

<https://www.oreilly.com/library/view/get-programming-with/9781617294204/kindle_split_045.html#ch30> (This is the Get Programming with ES Next book; it uses the axios module found on the npm.)

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function> on asynch functions – which return a promise