**Promises**

Promises are one of the most important topics in JavaScript, and thanks to a student of mine (Misa) for asking the question.  This brief article will help you understand the basics of Promises.

What is a **promise**?

In JavaScript, a Promise object represents an asynchronous event or result that will eventually happen (or not). A promise may be in one of 3 possible states: fulfilled, rejected, or pending.

Graphical user interface, text

Description automatically generated

But lets’ get real. This doesn’t make much sense, at least not to me when I first started out with promises.

So let me explain.

**You're an actor**

Imagine you’re an actor, and fans keep asking you about when your next movie will be released.

To keep the fans happy, you **promise**to invite them to a premiere when it’s done. You give your fans a contact list to fill in. Now, when you’re finished shooting the final scene and the editing team has given the green light, all of your fans will receive the invite, instantly. If the movie is never released for any reason, you can also notify them of the bad news.

This sounds great, right? You’re happy because you're looking after your fans. And the fans are happy because they know they will be the first in line to get notified (whether it’s a good result or not).

And this, my dear students, is how JavaScript promises work.

**A little more to promises**

**When we code, we often have code that needs to do “something”.** Often this “something” takes time to complete (for example, saving user data to a database or retrieving user information). This “something” is finishing the movie in our analogy above.

But this is not all.

Often we also have code that relies on the result of the code above. For example, maybe you want to display a successful message to a user, but only once the query to the database is done and successful. Many other functions in your code may also rely on the result of your first block of code doing “something”. These functions are the instant notifications sent to the fans in our analogy above.

**So Clyde, what is a promise?**

You can think of a JavaScript promise is a bridge.

It bridges the gap between your first block of code that needs to do “something” (in our example, finishing a movie) and later functions that rely on the first block of code being complete (in our example, sending an instant message). The bridge between these two is what? That’s right, its the **promise**. In our example above, it’s the contact list.

Think about it.

The contact list (the promise), makes it possible to send the result (was the movie released or not) to all the fans.

**Promises are more complex**

The analogy above is a simple one.

In practice, JavaScript promises are more complex – they have additional features that we haven’t spoken about.

But the analogy above should give you a high level understanding on what a promise is.

But if you’re anything like me, you want to at least see how it works in practice with an example.

Okay then.

A Promise is usually created via a constructor. Usually like this: let promiseObject = new Promise(executor);

**IMPORTANT: the word Promise is a special keyword defined in the JavaScript language and we did not have to create it. It’s available  straight out of the box.**

When new Promise is created, the executor function runs automatically. Its arguments are resolve and reject. The meaty code we write sits inside this executor function.

When the executor function gets the result, it will then call one of these 2 callbacks:

1. resolve(value) — if the call finished successfully
2. reject(error) — if an error occurred

**Let's see a promise in action**

Lets do a quick example and I’ll leave it here for you to dig deeper.

Lets say you are at an online casino. And all you need to do is roll the digital dice. If you get a number greater than 3, then you're a winner. Less than 3 and boo hoo ... you're a loser.

First lets create our Promise, and then lets call it. Here's the code:

Text

Description automatically generated

*For those of you who don't want to write out the code, I've attached the .html file.*

**Conclusion**

I know there’s a lot to this, and it can seem overwhelming. There is also the then and catch keywords which we haven’t spoken about.

For now, just remember:

* A promise is a JavaScript object that produces a single value, some time in the future
* A pending promise will eventually transition into a fulfilled or rejected state
* The executor function (our meaty, juicy code) can only call one resolve or one reject. All others (if you have them in there) are simply ignored
* Every promise supplies a .then() method
* Because .then() always returns a new promise, it’s possible to chain promises. We often chain a .catch() method to the end of a chain, to handle any errors that our promise throws back at us. This is what I did in the example code.

**Hope this guides and gets you pumped to learn more about promises.**

See you in the next lecture.