Promise

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Promises

* A promise is a object that represents and imtermidiate state of object.
* There is no guarantee of exactly when the operation will complete and the result will be returned, but there is a guarantee that when the result is available, or the promise fails, the code you provide will be executed in order to do something else with a successful result, or to gracefully handle a failure case.
* One of the most common engagements you'll have with promises is with web APIs that return a promise.
* Example :

chooseToppings()

.then(toppings =>

placeOrder(toppings)

)

.then(order =>

collectOrder(order)

)

.then(pizza =>

eatPizza(pizza)

)

.catch(failureCallback);

* Real time Example :

1. Fetch image form url it return promise store to promise.

let promise = fetch('coffee.jpg');

1. To respond to the successful completion of the operation whenever that occurs (in this case, when a [Response](https://developer.mozilla.org/en-US/docs/Web/API/Response) is returned), we invoke the [.then()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/then) method of the promise object. The callback inside the .then() block runs only when the promise call completes successfully and returns the [Response](https://developer.mozilla.org/en-US/docs/Web/API/Response) object — in promise-speak, when it has been **fulfilled**. It is passed the returned [Response](https://developer.mozilla.org/en-US/docs/Web/API/Response) object as a parameter.
2. We immediately run the blob() method on this response to ensure that the response body is fully downloaded, and when it is available transform it into a Blob object that we can do something with. The result of this is returned like so:

response => response.blob()

1. Unfortunately, we need to do slightly more than this. Fetch promises do not fail on 404 or 500 errors — only on something catastrophic like a network failure. Instead, they succeed, but with the [response.ok](https://developer.mozilla.org/en-US/docs/Web/API/Response/ok) property set to false. To produce an error on a 404, for example, we need to check the value of response.ok, and if false, throw an error, only returning the blob if it is true. This can be done like so — add the following lines below your first line of JavaScript.

let promise2 = promise.then(response => {

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

} else {

return response.blob();

}

});

1. Each call to .then() creates a new promise. This is very useful; because the blob() method also returns a promise, we can handle the Blob object it returns on fulfillment by invoking the .then() method of the second promise. Because we want to do something a bit more complex to the blob than just run a single method on it and return the result, we'll need to wrap the function body in curly braces this time (otherwise it'll throw an error).

Add the following to the end of your code:

let promise3 = promise2.then(myBlob => {

})

fetch('coffee.jpg')

.then(response => {

if (!response.ok) {

throw new Error(`HTTP error! status: ${response.status}`);

} else {

return response.blob();

}

})

.then(myBlob => {

let objectURL = URL.createObjectURL(myBlob);

let image = document.createElement('img');

image.src = objectURL;

document.body.appendChild(image);

})

.catch(e => {

console.log('There has been a problem with your fetch operation: ' + e.message);

});