**WebDevelopment II – Lab #1**

**Objective:** In this lab we will start to practice promises and fetch API .

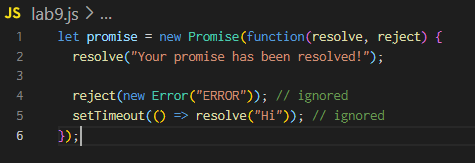
1. What are promises? What are the different states of a promise.

A promise is an object that may produce a single value sometime in the future: either a resolved value, or a reason that it’s not resolved (e.g., a network error occurred). A promise may be in one of 3 possible states: fulfilled, rejected, or pending. Promise users can attach callbacks to handle the fulfilled value or the reason for rejection. Promise can be rewrote with Async/Await function. It is used to achieve asynchronous operation and is a replacement of callback hell

1. What are the 2 parameters inside the callback functions that the promise returns?

Resolve and reject

1. How do you use promises? Give an example.



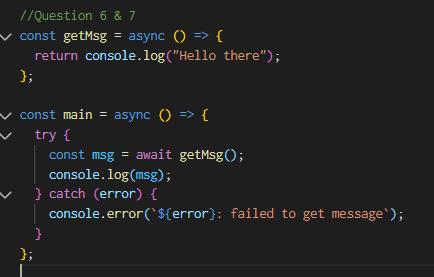
1. What is the difference between asynchronous and synchronous?

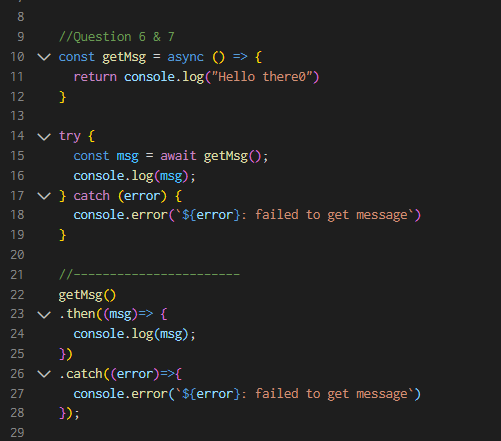
synchronous code is executed in sequence – each statement waits for the previous statement to finish before executing. Asynchronous code doesn’t have to wait – your program can continue to run. You do this to keep your site or app responsive, reducing waiting time for the user.

1. How do we catch errors for promises?

Use .catch() to the end of chain. Alternatively throw new Error()

1. How do you write a try-catch block?
2. Try to convert from .then() => async await





1. Give me 2 examples in a project when you want to use promise

When you do api call and when you need to chain multiple callback function to keep the order of execution.

OUTPUT QUESTIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 ) What is the output of the following code

const promise = new Promise((resolve, reject) => {

reject(Error('Some error occurred'));

})

promise.catch(error => console.log(error.message));

promise.catch(error => console.log(error.message));

* Some error occurred
* Some error occurred

2 ) const promise = new Promise((resolve, reject) => {

reject(Error('Some Error Occurred'));

})

.catch(error => console.log(error))

.then(error => console.log(error));

* Some error occurred

3) async function func() {

return 10;

}

console.log(func());

* 10

4) async function func() {

await 10;

}

console.log(func());

* Pending (promise won’t be fullfilled)

5) function delay() {

return new Promise(resolve => setTimeout(resolve, 2000));

}

async function delayedLog(item) {

await delay();

console.log(item);

}

async function processArray(array) {

array.forEach(async item => {

await delayedLog(item);

})

}

processArray([1, 2, 3, 4]);

* Need async keyword to execute this function.  
  After adding the async keyword, the output will be after 2 seconds, 1->2->3->4 will be logged one by one