**Assignment no 5(Async & Await)**

1. Create a function called "fetchFast" that returns a Promise that resolves with the String "Fast Done!" after 2 seconds

2. Create a function called "fetchSlow" that returns a Promise that resolves with the String "Slow Done" after 6 seconds

3. Print out "Program starting...

" 4. Create an async function that uses await to wait for the data to comes back from "fetchFast" then log out the data. Then use await to wait for the data to come back from "fetchSlow" and log it out right after. 5. Call the async function you created

6. Log out "Program complete!"

7. How long does this take and why?

8. How could you speed it up? using html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Async Await with Multiple Promises</title>

</head>

<body>

    <script>

        // Step 1: Create fetchFast function

        function fetchFast() {

            return new Promise((resolve) => {

                setTimeout(() => {

                    resolve("Fast Done!");

                }, 2000);

            });

        }

        // Step 2: Create fetchSlow function

        function fetchSlow() {

            return new Promise((resolve) => {

                setTimeout(() => {

                    resolve("Slow Done");

                }, 6000);

            });

        }

        console.log("Program starting..."); // Step 3

        // Step 4: Create async function to await fetchFast and fetchSlow in sequence

        async function startProgram() {

            const fastResult = await fetchFast(); // Wait for fetchFast to resolve

            console.log(fastResult); // Log the result of fetchFast

            const slowResult = await fetchSlow(); // Wait for fetchSlow to resolve

            console.log(slowResult); // Log the result of fetchSlow

        }

        startProgram(); // Step 5

        console.log("Program complete!"); // Step 6

    </script>

</body>

</html>

7.The total time taken for this code is approximately **8 seconds**:

* fetchFast takes 2 seconds, and only after it completes does fetchSlow begin, taking another 6 seconds.

8. To make it faster, we can start both promises simultaneously and then await each one as soon as it's ready. Using Promise.all() lets us run them concurrently:

javascript

Copy code

async function startProgram() {

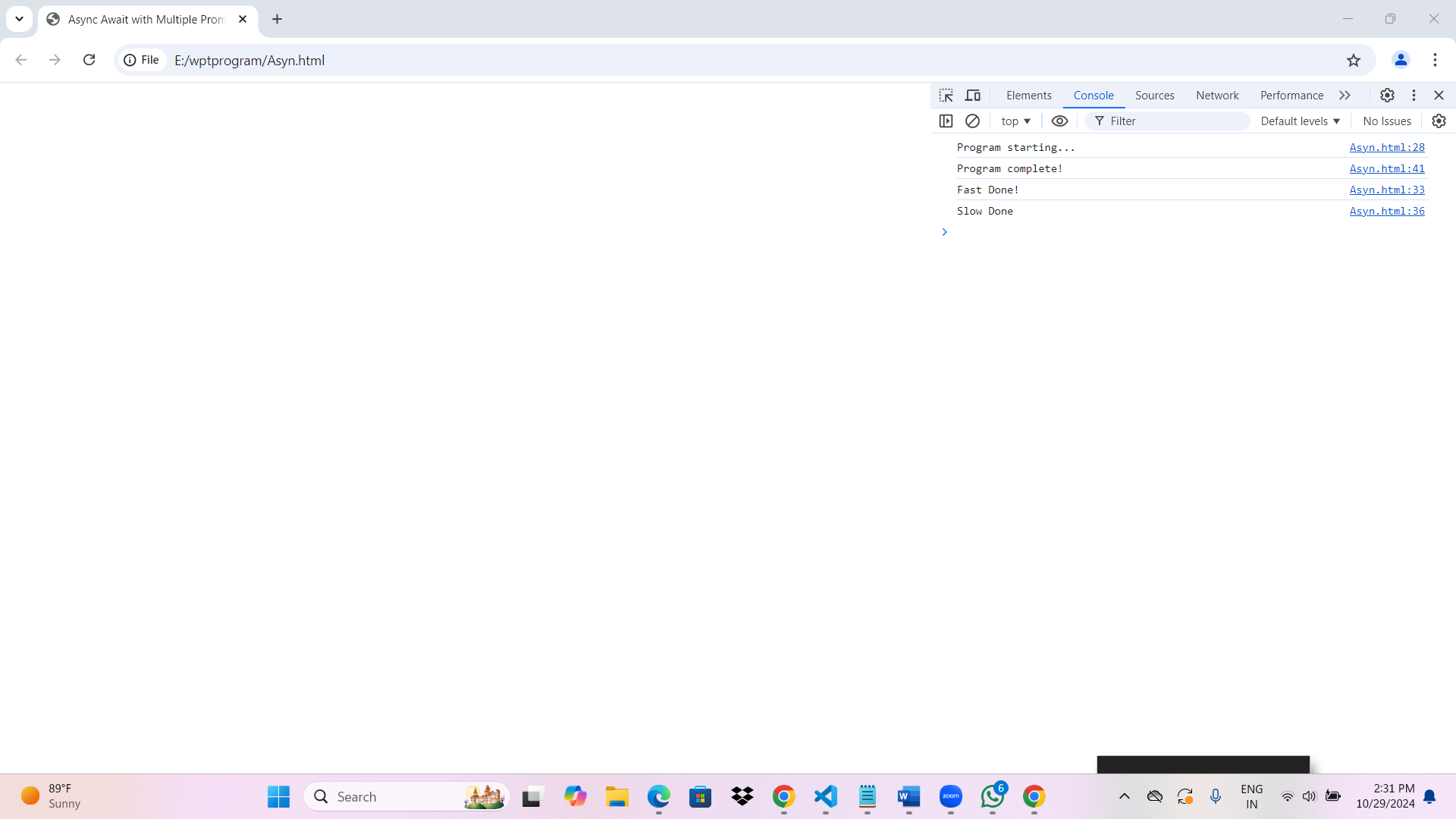
const [fastResult, slowResult] = await Promise.all([fetchFast(), fetchSlow()]);

console.log(fastResult);

console.log(slowResult);

}

With Promise.all(), both promises run in parallel, reducing the time to the longest promise duration, **6 seconds**.



Q.2.

1. Create a function called "goGetCandies" which will return a Promise Object that resolves to the value: { candy: "sour keys", quantity: 10 } This should take 2 seconds

2. Create another function called "sellCandies" that will take a candy Object like above ^ and return a Number that is 25 \* quantity. This will be how much (in cents) we get for our candies. However, make this function take 3 seconds to do this math (return a Promise with a setTimeout with the answer).

3. Write an async function that uses await to: 1. Get the candy object from goGetCandies() 2. Passes it to "sellCandies" and waits for the response 3. Prints out how much money we made from our sale

4. Do the same steps as #3 but with vanilla Promises.

* **Q1: Which of these 2 methods do you prefer?**  
  *The async/await version is generally preferable for handling sequential tasks in a more readable way.*
* **Q2: Which of these 2 methods is easier to read?**  
  *The async/await version is typically easier to read since it resembles synchronous code structure, making it clearer to understand each step.*
* !DOCTYPE html>
* <html lang="en">
* <head>
* <meta charset="UTF-8">
* <meta name="viewport" content="width=device-width, initial-scale=1.0">
* <title>Async vs Promises</title>
* </head>
* <body>
* <script>
* // Step 1: Create goGetCandies function
* function goGetCandies() {
* return new Promise((resolve) => {
* setTimeout(() => {
* resolve({ candy: "sour keys", quantity: 10 });
* }, 2000);
* });
* }
* // Step 2: Create sellCandies function
* function sellCandies(candyObj) {
* return new Promise((resolve) => {
* setTimeout(() => {
* resolve(25 \* candyObj.quantity);
* }, 3000);
* });
* }
* // Step 3: Async/await version
* async function sellCandiesWithAsync() {
* const candy = await goGetCandies();
* const money = await sellCandies(candy);
* console.log(`Money earned (Async): ${money} cents`);
* }
* sellCandiesWithAsync();
* // Step 4: Vanilla Promises version
* function sellCandiesWithPromises() {
* goGetCandies()
* .then((candy) => sellCandies(candy))
* .then((money) => {
* console.log(`Money earned (Promises): ${money} cents`);
* });
* }
* sellCandiesWithPromises();
* </script>
* </body>
* </html>

