**Assignment no 5(promise)**

***1. Print out "Program started" at the start of your code***

***2. Create a Promise that resolves after 5 seconds with the value {data: "Hello, friend!", error: null}***

***3. Log out the promise while it's pending***

***4. Print out "Program in progress..." as well***

***5. Create a first Promise chain using the promise above and Print out the resolved value when the first promise fulfills***

***6. Have this first promise return another new Promise that will fulfill after 2 seconds with the message: "First promise chain complete!"***

***7. Print out the message from the above promise after it fulfills ("First promise chain complete!")***

***8. Create a second Promise chain using the first promise above and Print out the resolved value when the second promise fulfills***

***9. Have this second promise return another new Promise that will fulfill after 10 seconds with the message: "Second promise chain complete!"***

***10. Print out the message from the above promise after it fulfills ("Second promise chain complete!")***

***HINT: Use setTimeout for the delay HINT2: This will be using the same promise two times: const myPromise = new Promise(...)***

***// step 2 myPromise.then(...).then(...) //***

***steps 5-7 myPromise.then(...).then(...) //***

***steps 8-10 BONUS: WHY does it work this way?***

***<!DOCTYPE html>***

***<html lang="en">***

***<head>***

***<meta charset="UTF-8">***

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

***<title>Promise Chains</title>***

***</head>***

***<body>***

***<script>***

***console.log("Program started"); // Step 1***

***// Step 2: Create a Promise that resolves after 5 seconds***

***const myPromise = new Promise((resolve, reject) => {***

***setTimeout(() => {***

***resolve({ data: "Hello, friend!", error: null });***

***}, 5000);***

***});***

***console.log(myPromise); // Step 3: Log the promise while it's pending***

***console.log("Program in progress..."); // Step 4***

***// First Promise chain***

***myPromise***

***.then(result => {***

***console.log(result); // Step 5: Print out resolved value***

***return new Promise((resolve, reject) => {***

***setTimeout(() => {***

***resolve("First promise chain complete!"); // Step 6***

***}, 2000);***

***});***

***})***

***.then(message => {***

***console.log(message); // Step 7: Print out message from above promise***

***});***

***// Second Promise chain***

***myPromise***

***.then(result => {***

***console.log(result); // Step 8: Print out resolved value***

***return new Promise((resolve, reject) => {***

***setTimeout(() => {***

***resolve("Second promise chain complete!"); // Step 9***

***}, 10000);***

***});***

***})***

***.then(message => {***

***console.log(message); // Step 10: Print out message from above promise***

***});***

***// BONUS Explanation:***

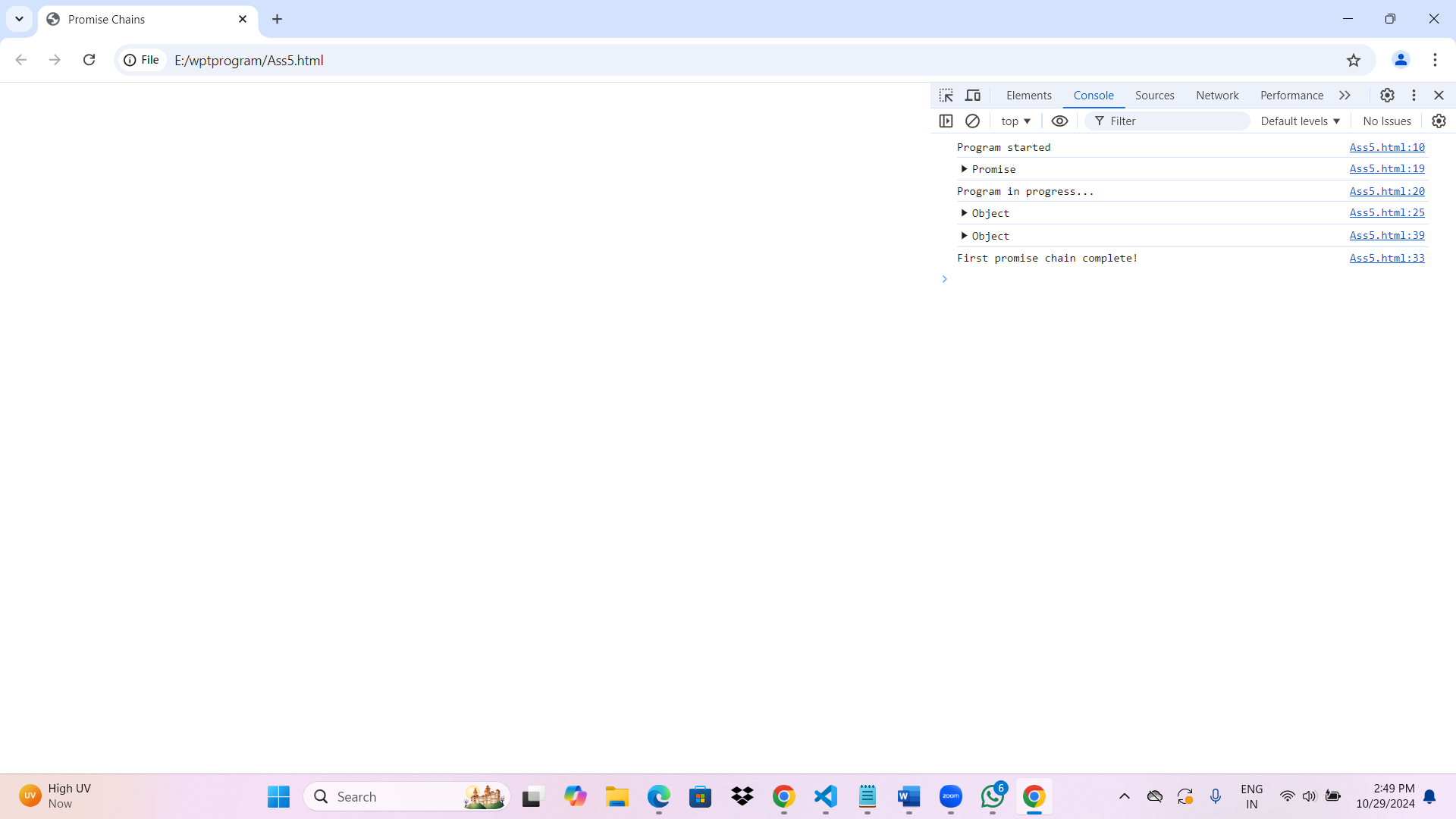
***// This works because `myPromise` resolves only once, and it’s re-used in both chains.***

***// Once a promise has resolved, calling `.then()` on it returns a resolved promise with that value.***

***</script>***

***</body>***

***</html>***

******

***Why it Works This Way:***

***The promise myPromise resolves once, so calling .then() multiple times still provides access to the resolved value. Each .then() returns a new promise, allowing us to build separate chains based on the same initial promise.***