Q1 Is JavaScript single threaded or multithreaded? What does it mean to be any? \*

JavaScript is single threaded language,

JavaScript can only run one instruction at a time,

while multithreaded languages could run multiple instructions concurrently.

Q2 What are promises? Why are they used? \*

Promises in JavaScript represent processes that are already happening,

which can be chained with callback functions.

Promises are used to handle asynchronous operations in JavaScript.

They are easy to manage when dealing with multiple asynchronous operations

where callbacks can create callback hell leading to unmanageable code.

Q3 What do async/await do? Explain it in your own words. \*

Async functions will always return a value.

It makes sure that a promise is returned and if it is not returned then

javascript automatically wraps it in a promise which is resolved with its value.

Await function is used to wait for the promise.

It could be used within the async block only.

It makes the code wait until the promise returns a result.

It only makes the async block wait.

Q4 How do we catch errors in async functions? \*

 common way to handle errors when awaiting a promise is to wrap it with a try/catch block.

Q5 What do async functions return? \*

Async functions always return a promise.

Q6 What do then() consumers in promises return? \*

then() method returns a new Promise that immediately resolves to the return value.

Q7 Write the explicit equivalent of Promise.resolve(100);

p1 = Promise.resolve(50);

p2 = 200

p3 = new Promise(function(resolve, reject) {

setTimeout(resolve, 100, 'solved');

});

Promise.all([p1, p2, p3]).then(function(values) {

document.write(values);

});

Q8 What are the states a promise can be in? \*

pending - The initial state of a promise.

fulfilled - The state of a promise representing a successful operation.

rejected - The state of a promise representing a failed operation.

Q9 What happens if neither resolve nor reject is called within a promise? \*

Promise always remains in pending state

Q10 What happens if multiple resolve or reject is called within a promise? \*

Only the first resolve or reject will be executed remaining will be ignored

Q11 What does the finally() method on promise do? Provide your explanation.

The finally() method returns a [Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise). When the promise is settled, i.e either fulfilled or rejected, the specified callback function is executed. This provides a way for code to be run whether the promise was fulfilled successfully or rejected once the Promise has been dealt with.

Q12 What are microtasks in JS? \*

A microtask is **a short function which is executed after the function or program which created it exits** and only if the JavaScript execution stack is empty, but before returning control to the event loop being used by the user agent to drive the script's execution environment.

Q13 Write a simple print() function that accepts a callback, which gets called after 3 seconds. The callback function could be anything that prints out on the screen. \*

const print = function() {

alert("This message is shown after 3 seconds");

}

setTimeout(print, 3000);

Q14 Delay with a promise We know that setTimeout() uses a callback.Create a promise based alternative.The function named delay(ms) should return a promise.We should be able to achieve something like this:delay(3000).then(() => alert('Should alert after 3 seconds’)

function delay(milisec) {

return new Promise(resolve => {

setTimeout(() => { resolve('') }, milisec);

})

}

delay(3000).then(() => alert('Should alert after 3 seconds'));

Q15 Consume the following promise using `await`.let promise = new Promise((res,rej) => { setTimeout(() => res("Resolved!"),3000)} );promise.then((res) => alert("Response value is: " + res)); // this line needs to get replaced.

async function func() {

let promise = new Promise((res, rej) => {

setTimeout(() => res("Resolved!"), 3000);

});

let result = await promise;

alert(result);

}

func();

Q16: What is the difference between the following two lines of code:promise.then(f1).catch(f2);and promise.then(f1, f2);Provide an explanation for your answer.

In promise.then(f1).catch(f2)

 if an error happens in f1, then it is handled by .catch

In promise.then(f1, f2);

No catch block to handle the error