**1. What is the difference between synchronous and asynchronous code?**

Synchronous code is a number of lines that are executed in sequence one after the other in order that they are written. For example:

const firstName = “James”

const secondName = “Doe”

const fullName = `${firstName} ${secondName}`

console.log(fullname)

Here, the code starts at the top and will be executed 1 line at a time in order until it reaches the end.

Asynchronous code on the other hand is code that will not necessarily run in sequential order (as the above code would), due to a number of reasons such as invoking a call to a 3rd party API and having to wait for the response to return. In the meantime, the code sitting behind that call will, if not handled correctly, go ahead and run synchronously. For example:

console.log("Start Code");

setTimeout(() => {

  console.log("Waiting for timeout to complete....");

}, 3000);

console.log("Finish Code")

The result of running this code will be:

**Start Code**

**Finish Code**

**Waiting for timeout to complete....**

This is obviously not what we wanted to happen (in this case), however in other cases, it is exactly what we want to happen, for example when loading db data into a page, we want the remainder of the page to render in the meantime so that the page isn’t sitting there half complete waiting for the data to be returned.

**2. Is javascript synchronous or asynchronous?**

Read here: <https://ankitkamboj18.medium.com/is-javascript-synchronous-or-asynchronous-what-the-hell-is-a-promise-302ee008dfcd>

At its core, JavaScript is a synchronous, single threaded language processing one operation at a time. However, Javascript can be manipulated to behave asynchronously.

**3. What is a promise in js?**

Read: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises>

Given that we won’t necessarily know the order in which functions/code dependencies will be returned in we cannot simply rely on them coming back to us in the order we want them to. Therefore, we can separate out each of the dependencies, sending off the asynchronous function and then hold back the remaining dependencies until it has returned. this is known as a promise.

The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

**4. How can you handle promises in js?**

const promise = doSomething();

const promise2 = promise.then(successCallback, failureCallback);

5. What do the Async/Await keywords do in js?

Read: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Async_await>

const hello = async() =>{

  return "hello"

}

// this returns a promise

console.log(hello())

// this consumes the promise and returns the value of it

hello().then((value)=> console.log(value))

//can also be written

hello().then(console.log)

// await

const goodbye = async() =>{

  return await Promise.resolve("goodbye")

}

goodbye().then(console.log)