Promises:

The biggest differences between a promise() and an Await/async function is the scope over which they cover. Promises literally only cover their own little promise chain, and nothing else within the function. So for example, if we had the following code snippet:

const myFunction = (data) => {

returnsAPromise(data)

.then(res => {

//action of some type on data

});

console.log(“This is still running”);

}

myFunction(myData);

In this example, the only blocked or asynchronous part is the actual call to the function that returns a promise. The example will print “This is still running” before the promise finishes resolving. You can also manage promises better in parallel will the Promise.all() call to basically reap all of the outstanding promises.

Async/Await:

With async/await functions the whole function is awaited, and the process is not continued until the promise is fulfilled, so, using the same code snippet, but now modified to be an async/await call

const myFunction = async (data) => {

const results = await returnsAPromise(data)

//action of sometype on results

console.log(“This isn’t still running”);

}

await myFunction(myData);

Here the string will NOT print out until the function finishes resolving and performs its own action. The await call to the function itself is also blocking until the entire function finishes running. In my experience, I have mostly used Await/async functions to ensure that the call to an api/database has resolved before I try and use its result for anything. But the promises themselves, are generally written in libraries I call to. Promises are still essential to understand however, and these little code snippets displayed one of the principle differences they have.