Async programming and promises

What we know :

Js is single threaded language.

Single threaded language means it executes one thing at a time.

So in a single threaded language problem arises when you r doing an extensive task. Then that extensive task would block the thread. You you'll run into a problem. Because rest of the code is not executing.

Extensive task - heavy task that takes a lot of processing for example you r fetching data from database or loop running till 10000000.

NOW WHAT WE KNOW…..

PROBLEM ????????

This problem is solved by asyncronronous programming pattern in javascript.

For that we should know.

Promises -> resolve, reject callbacks.

So PROMISE —----------->

PROMISE HAS TWO RESULTS —--> RESOLVED / FULLFILLED OR REJECT / UNSUCCESSFUL

A promise is a wrapper in which you write your blocking code(code that would block the thread).

You now create a promise 🎉

```
    const myPromise = new Promise((resolve, reject) => {
     //                          //                    //
//  here write logic in such a way that it will be either resolved / successful or rejected / failed

////                        //

});
```

```
Let p = new Promise((res, rej) =>{

   Let a  = 1 +90;
   if(a == 2){
    resolve('success')
   }else{
    reject('failed')
   }
 })
```

Another way of doing async programming.

```
function resolveAfter2Seconds() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000);
  });
}

async function asyncCall() {
  console.log('calling');
  const result = await resolveAfter2Seconds();
  console.log(result);
  // Expected output: "resolved"
}

asyncCall();
```