Promise in javascript

Promise in javascript is an object by which we can ensure some task to be done on a specific ensurity by handling error and handle everything by saving the code from callback hell.

we can work with promises with two ways:-

(I) by making a new object and resolve reject into it

(II) by using a function and passing parameters into it and handling the data into it with the help of data passed by other promise.

promise is handled with the help of 3 steps:-

[(I) pending → (II) Result → (IV) display]

While working with promises you will land on one situation atleast for sure without any hills like calling a function in a loop and promise holds a priority in the queue of execution in javascript.

```
const promise = new Promise ((resolve, reje
{

});
```

Async/Await

⇒ There is a special syntax to work with promises in a more comfortable fashion, called "(async/await)". It's surprisingly easy to use.

⇒ The word "async" before a function means one simple thing: a function always returns a promise.

⇒ So the async keyword is added to functions to tell them to return a promise rather than directly returning the value.

⇒ We can use await when calling any functions that returns a promise, including web API functions.

⇒ The key word await makes Javascript wait until that promise settles and ~~var~~ returns its result.

⇒ If we attach async with a function and get response

```
try{ Async function Reminder(){
Var num = await <Promise name>
Console.log(num) }
return num; }.Catch(error){
Console.log(error)}
```

⇒ This can be used for information Retaining

Fetch APIs

Fetch API Method ⇒ fetch() [This is a part of modern AJAX]

This is also a promise which works on dependency of then, Catch and finally Iterations and provides result.

In modern JS we use fetch() otherwise old times, we used XMLhttp req method to fetch apis from Server.

```
fetch('url').then(() => {
    Some Conditions / Conversions / Coursions
    return Something on which next then
}).then(() => {
    behaviour depends on
    previous then
}).Catch
(erro
```

And at last we use finally.

Call, Apply & Bind methods

(1) Call method

⇒ It is a Simple way by help of this we can use one object's method in another object.

An object can have different methods and we should just write them only once and use it any where in the program.

Apply method

⇒ Apply method is Similarly Same but in Apply method we pass argument in arrays.