**JavaScript Promises:**

Js is predominantly single threaded. This makes it slow and restrictive.

With the help of Promises and other Asynchronous concepts, Js can perform long network requests simultaneously without blocking the main thread.

What are Js Promises?

A Promise is an asynchronous action that may complete at some point in the future and reduce a value.

Promises provide a robust way to wrap the result of asynchronous work, overcoming the problem of deeply nested callbacks.

**States of Promises:**

**1. Pending:**

The underlying operation not yet completed.

**2. Fulfilled:** The Operation has finished.

3. **Rejected:**

An error has occurred during the operation, and the promise is rejected.

A promise is said to be settled either fulfilled or rejected.

Once a promise is settled, it becomes immutable, and its state cannot change.

then(callback) – used to attach a call back when the promise is resolved/fulfilled.

catch(callback) – Used to attach a callback when the promise is rejected.

**General Syntax:**

Let someaction = new promise(function(resolve,reject) {

//Perform some works

})

Example1: Promise

<html>

<head>

    <title> JS Promises </title>

    <p id="demo"></p>

</head>

<body>

    <script>

        let car = new Promise(fuction(resolve,reject)){

            fuel\_fulltank = true

            if(fuel\_fulltank)

            resolve

            else

            reject()

        });

        car.then(function(){

            document.write("The fuel tank is full!Happy Driving!");

        }).catch(fuction()){

            document.write("The fuel tank is not empty");

        </script>

</body>

</html>

Example2: Nested Promise

//Nested Promises

<!DocType html>

<html>

    <head>

    </head>

    <body>

        <script>

            let empty\_tank = function(){

                return new Promise(function(resolve,reject){

                    resolve("The car doesnot have enough fuel")

                })

            }

            let engine = function(){

                return new Promise(function(resolve,reject){

                    resolve("The engine is over heating")

                })

            }

            let travel = function(){

                return new Promise(function(resolve,reject){

                    resolve("The car is not safe for travelling")

                })

            }

            empty\_tank().then((function(){

                return engine(result)

            }).then (function(result){

                return travel(result)

            }).then (function(result) {

                console.log("Done"+result)

            })

            </script>

    </body>

</html>

**JS Async and Await:**

**Synchronous** – process Executed one by one.

Start

Function1 – wait (resource)

Function2 – After Execution of the first process, the second one will be executed.

Function3 – After function2, Function3 Executed.

**Asynchrouns** – Don’t halted for any other process, it execute the process simultaneously.

Start

Fn1 - wait (resource)

Fn2 – function2 is running, it won’t wait for the function1.

Fn3 – function3 is running after function2.

**Async and Await:**

Async and Await is used in the promise to do the Asynchronous function with await. Await means it wait for the response from the Promise object.

Example: async and await

//Async and Await

<!DocType html>

<html>

    <head>

    </head>

    <body>

        <script>

            let result = function(score) {

                return new Promise(function(resolve,reject){

                    console.log("calculating results...")

                    if(score>50)

                    resolve("congratulation!you have Passed")

                    else

                    reject("you have failed")

                })

            }

            let grade = function(response) {

                return new Promise(function(resolve,reject){

                    console.log("calculating your grade...")

                    resolve("Your grade is A"+response)

                    else

                    reject("No Grade")

                })

            }

            result(30).then(response => {

                console.log("Results received")

                return grade(response)

            }).then(finalgrade=>{

                console.log(finalgrade)

            }).catch(err=>{

                console.log(err)

            })

            //async instead .then

            async fuction calcluateResult() {

                try{

                    const response = await(result(80),setTimeout(function())

                    console.log("Results Received")

                    const finalgrade = await grade(response)

                    console.log(finalgrade)

                }

                catch(err)  {

                    console.log(err)

                }

            }

            calcluateResult()

            </script>

    </body>

    </html>