**S6\_\_Promise & Async Await**

JavaScript ES6 (also known as ECMAScript 2015 or ECMAScript 6) is **the newer version of JavaScript that was introduced in 2015**. ECMAScript is the standard that JavaScript programming language uses. ECMAScript provides the specification on how JavaScript programming language should work.

**ES6 Features :-**

Default parameters

Template literals (Template strings)

Tagged Templates

Destructuring assignment

Arrow function expressions

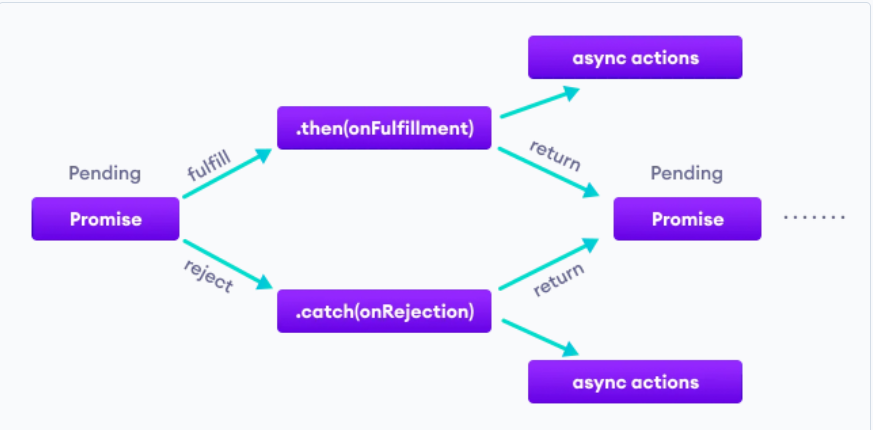
let and const

Spread and Rest syntaxes (…)

Object.assign() and Object.is()

Classes

**Promise** :- are a way to implement asynchronous programming in JavaScript(ES6 which is also known as ECMAScript-6). A Promise acts as a container for future values. For asynchronous programming, JavaScript uses [callbacks](https://www.geeksforgeeks.org/javascript-callbacks/) , but there is a problem using the callback which is callback hell (multiple or dependent callbacks) or Pyramid of Doom. Using the ES6 Promise will simply avoid all the problems associated with the callback.



**Promises has having states but only one state it will consider.**

* **fulfilled:** Action related to the promise succeeded.
* **rejected:** Action related to the promise failed.
* **pending:** Promise is still pending i.e not fulfilled or rejected yet.
* **settled:** Promise has fulfilled or rejected.

**Callback to Promise : Types**

* **.then():** Invoked when a promise is kept or broken. It can be chained to handle the fulfillment or rejection of a promise. It takes in two functions as parameters. The first one is invoked if the promise is fulfilled and the second one(optional) is invoked if the promise is rejected. Example: Handling Promise rejection using .then()
* .**catch ()** :- can be used for handling the errors(if any). It takes only one function as a parameter which is used to handle the errors (if any). Example: Handling Promise rejection(or errors) using .catch()

//Promise

function promise() {

    return new Promise(function (resolve, reject) {

        setTimeout(() => {

            const error = true;

            if (!error) {

                console.log('Function: Your promise has been resolved')

                resolve();

            }

            else {

                console.log('Function: Your promise has not been resolved')

                reject('Sorry not fulfilled');

            }

        }, 2000);

    })

}

promise().then(function () {

    console.log("Resolved")

}).catch(function (error) {

    console.log("Bad Response" + error)

})

// //Example:-

const students = [

    { name: "Anis", subject: "Php" },

    { name: "Amit", subject: "Machine Learning" },

    { name: "Ashok", subject: "Python" },

    { name: "Ankit", subject: "Machine Learning" }

]

function enrollStudent(student) {

    return new Promise(function (resolve, reject) {

        setTimeout(function () {

            students.push(student);

            console.log("Student has been Enrolled");

            const error = true;

            if (!error) {

                resolve();

            }

            else {

                reject();

            }

        }, 1000);

    })

}

function getStudents() {

    setTimeout(function () {

        let str = "";

        students.forEach(function (student) {

            str += `<li> ${student.name}</li>`

        });

        document.getElementById('students').innerHTML = str;

        console.log("Students have been Fetched");

    }, 5000);

}

let newStudent = { name: "Imran", subject: "Python" }

enrollStudent(newStudent).then(getStudents).catch(function () {

    console.log("Some error occured");

});

getStudents();

// // function inside then is run  - resolve()

// // function inside catch is run  - reject()