Assignment 5b

Roll No : 71 Batch T14

Aim : Write a JavaScript program to implement the concept of Promise. Access the data from any API using Promise and either resolve or reject the request by taking appropriate action.

Theory:

Promises help manage asynchronous operations. They provide a way to handle asynchronous tasks in a more organized and readable manner, making it easier to work with asynchronous code.

Promises steps:

1. Creation: Promises are a way to handle asynchronous operations more effectively in JavaScript. You create a promise using the Promise constructor, which takes a function as its argument. This function is often referred to as the "executor."
2. Executor Function: The executor function is executed immediately when the promise is created. It takes two arguments: resolve and reject. Inside this function, you perform the asynchronous task you want to handle.
3. Resolve and Reject: When the asynchronous task is successful, you call the resolve function with the result. If an error occurs during the task, you call the reject function with an error message or object.

Handling Promises:

1. .then() Method: You can attach a .then() method to a promise to handle the resolved value. This method takes a callback function as an argument, which is executed when the promise is resolved. You can chain multiple .then() methods together to create a sequence of asynchronous operations.
2. Chaining Promises: Each .then() callback can return another promise, allowing you to chain asynchronous operations. This creates a clear sequence of steps to be executed as each promise resolves.
3. Error Handling: To handle errors, you can attach a .catch() method to a promise chain. This method catches any errors that occur in the promise chain and allows you to handle them gracefully.

Async/Await:

1. async/await: An alternative approach to working with promises is using the async keyword before a function declaration. Inside an async function, you can use the await keyword before a promise to pause execution until the promise resolves or rejects. This makes asynchronous code look more like synchronous code.
2. try/catch: When using await, you can wrap it in a try block to catch any errors that occur during the asynchronous operation.

Benefits:

Promises provide a more organized and readable way to handle asynchronous operations compared to deeply nested callbacks. They improve code structure, error handling, and make it easier to reason about asynchronous workflows.

Conclusion :

Promises manages asynchronous tasks. Created program to access the data from an API using Promise.Offer chained .then() for success and .catch() for errors, increases code readability and reliability.