**Promises** in Node.js are a way to handle asynchronous operations. Promises provide a cleaner and more structured way to handle asynchronous code compared to traditional callback-based approaches which was creating call back hell.They represent a placeholder for a value that might not be available yet, but will be resolved in the future, either successfully with a value or unsuccessfully with an error.

Here's a brief explanation of how promises work in Node.js:

**1. Creation:** You create a promise using the `new Promise()` constructor. Inside this constructor, you pass a function with two parameters: `resolve` and `reject`.

**2. States:** Promises have three possible states:

- Pending: Initial state, neither fulfilled nor rejected.

- Fulfilled: The operation completed successfully, and the promise now has a resolved value.

- Rejected: The operation failed, and the promise has a reason for the failure, typically an error.

Note: a promise as soon as it comes with a data, is called “resolved promise”. However, resolved promise might not always be successful, because there might be an error. So “”resolved promise can either be “fulfilled” or “rejected”

**3. Consumption:** You consume a promise using `.then()` and `.catch()` methods:

- `.then()`: This method is called when the promise is fulfilled. It takes a function as an argument, which will be executed when the promise is successfully resolved.

- `.catch()`: This method is called when the promise is rejected. It takes a function as an argument, which will be executed if the promise encounters an error.

**4. Chaining:** You can chain multiple promises together using `.then()` and `.catch()`. This allows you to perform a series of asynchronous operations sequentially.

--------------------------------------------------------------------------------------------------------------------------------

When using `async` and `await`, you can achieve asynchronous consumption in a more synchronous-looking manner. Here's how it replaces the `.then()` and `.catch()` methods:

**1. async Function Declaration:** You declare a function as `async` to be able to use the `await` keyword inside it. This allows you to pause execution until promises settle.

**2. await Keyword:** await is used to pause the execution of the `async` function until the promise is resolved. It allows you to write asynchronous code that looks and behaves more synchronously.

**3. Try-Catch Block:** Since `await` can be used only inside an `async` function, error handling is done with a try-catch block instead of `.catch()`.