2. Asynchronous Operation and Promises

Promises in Node.js provide a more readable and maintainable way to handle asynchronous operations compared to traditional callback methods. They help avoid “callback hell” (nested callbacks) and make the asynchronous code flow more linearly.

Here's a simple Node.js application that performs an asynchronous operation (reading a file) using promises:

```javascript
Const fs = require(‘fs’).promises;

// Function to read a file asynchronously using promises
Function readFileAsync(filePath) {
  Return fs.readFile(filePath, ‘utf8’);
}

// Example usage
Const filePath = ‘example.txt’;

readFileAsync(filePath)
  .then((data) => {
    Console.log(‘File content:’, data);
  })
  .catch((error) => {
    Console.error(‘Error reading file:’, error);
  });
```

Advantages of Promises over Callbacks:

1. **Readability:** Promises allow you to write asynchronous code in a more readable and sequential manner. With promises, you can chain `.then()` and `.catch()` calls, making the code easier to follow.

2. **Avoiding Callback Hell:** Promises help in avoiding the callback pyramid of doom or “callback hell” where multiple nested callbacks make the code difficult to understand. Promises allow you to flatten the structure and handle errors in a more organized way.

3. **Error Handling:** Promises provide a separate `.catch()` block for error handling, making it clearer and more consistent compared to mixing error handling with callbacks.

4. **Chaining:** Promises support chaining multiple asynchronous operations using `.then()`. This allows you to compose and sequence asynchronous tasks in a straightforward manner.

5. **Promise.all():** Promises provide the `Promise.all()` method, which allows you to wait for multiple asynchronous operations to complete before moving forward. This is more convenient than managing nested callbacks for parallel operations.

Overall, promises enhance code readability, maintainability, and error handling in asynchronous operations, leading to more maintainable and scalable Node.js applications.