2024

Node Js Task

JOLIANA EMAD KAMAL NAGUIB 20P3292

SAMSUNG INNOVATION CAMPUS

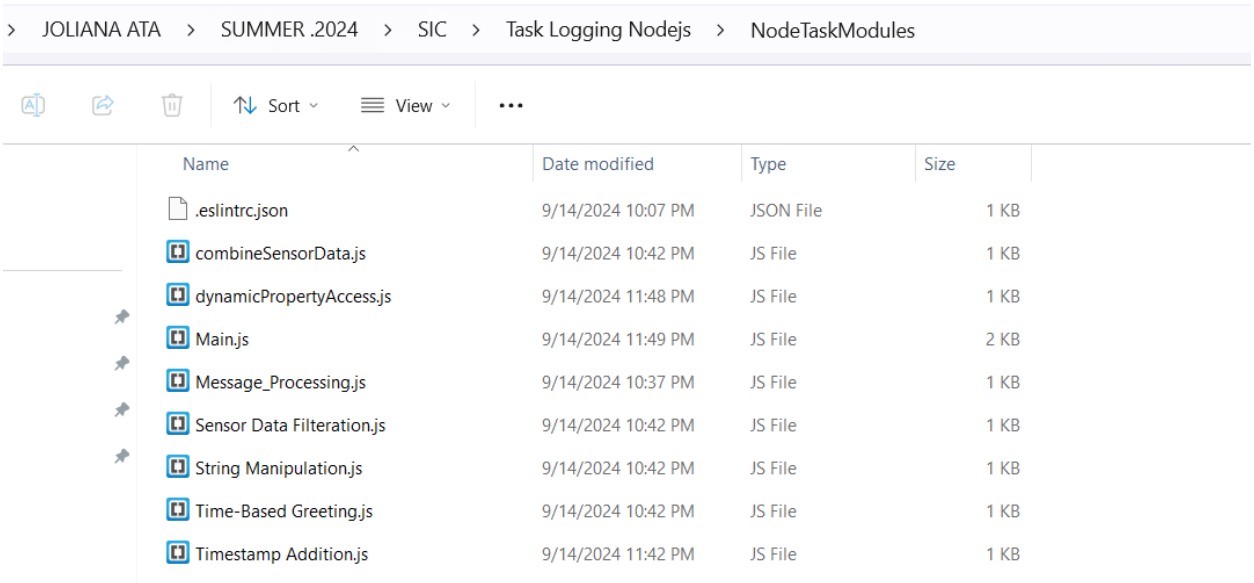
TASK Node.js

The Purpose of these tasks to be practice on Node js syntax and be familiar with many situations related to IoT fields.

It Helps you to write functions easily in Node Red and interact with i/p and o/p of nodes.

I divide my code into different modules to enhance organization, maintainability, and clarity.

‘The Main.js’ file serves as the central point where all functions are called from the various modules to execute each task separately.



1. **Message Processing**

   - Write a function that takes a JSON message (with `temperature`, `humidity`, and `device` properties) and adds a status based on the temperature (e.g., "High" if above 30°C).

Code:

How?

- The function first verifies that msg and msg.payload exist and that msg.payload contains a defined temperature. This stops errors related to undefined properties at the input level.

-If the input data is correct, the function extracts the temperature from msg.payload.temperature.

-Next, it determines if this amount exceeds thirty. It either assigns "High" or "Normal" to the status property in the msg.payload based on this situation.

-Finally, the updated status property of this changed message object is returned.

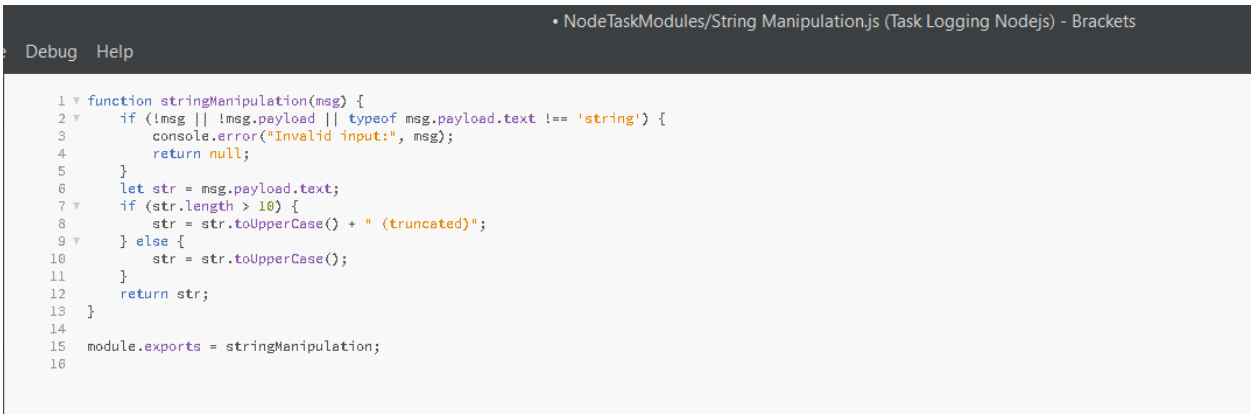
-The use of module.exports allows this function to be shared and used in other parts which is file ‘Main.js’ here.

Output:

2. **String Manipulation**

   - Create a function that converts a string to uppercase. If the string is longer than 10 characters, add `"(truncated)"` to the end.

Code:



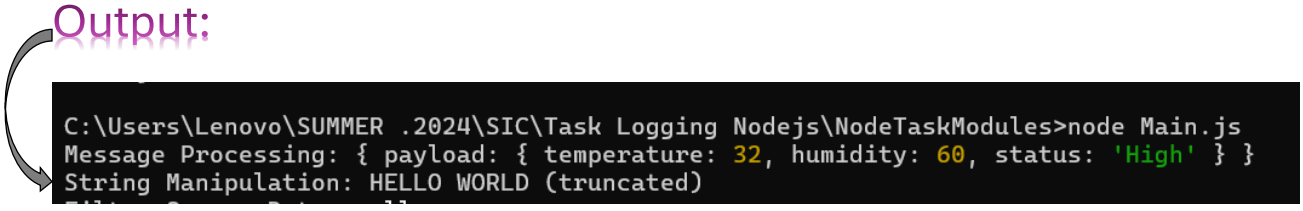
How?

- The goal of this function is processing a string of texts denoting whether they have more than ten letters or not by converting them all into capital letters and also adding a truncation notice to those who are beyond such limit.

-First of all in the processing stage, the function goes on and returns an input text from msg.payload.text if is valid. The length of the string is then checked; if it's over 10 characters, it will first be converted to uppercase and afterwards appended " (truncated)" at its end while for the other valid case, it will simply be converted into capital letters without any other additional information.

-The function returns the modified string. This result will either be a truncation informed uppercase string or just an upper case string (if it was ranging below ten characters).

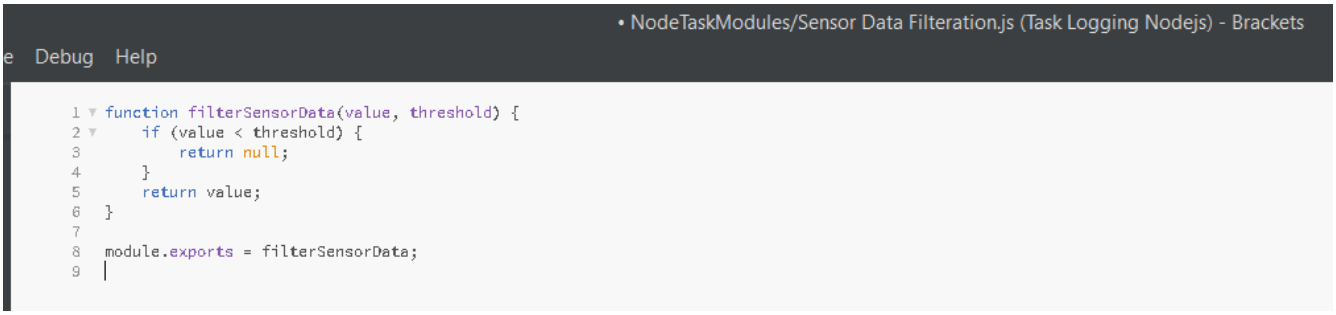
Output:



3. **Filter Sensor Data**

   - Write a function that filters out sensor readings below a certain threshold. Return `null` for values that should be ignored.

Code:



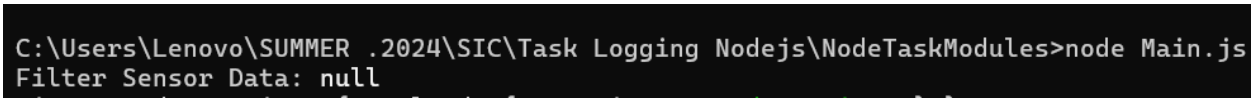
How?

This function filters out sensor readings that are below a specific threshold.
where Readings which are above or equal threshold can pass,

'value' which is the sensor reading which needs to be assessed.
&'threshold' is the minimum permissible amount. Any readings below this threshold are regarded as insignificant.
Processing:

The function compares the value with the threshold.
If the value is less than the threshold, it returns null. This shows that the reading does not fulfil the required minimum if it should be ignored or filtered out.
If the value is greater than or equal to the threshold, it returns the value itself. This implies that this reading is serious and may be vulnerable against such acts by anybody who controls it.

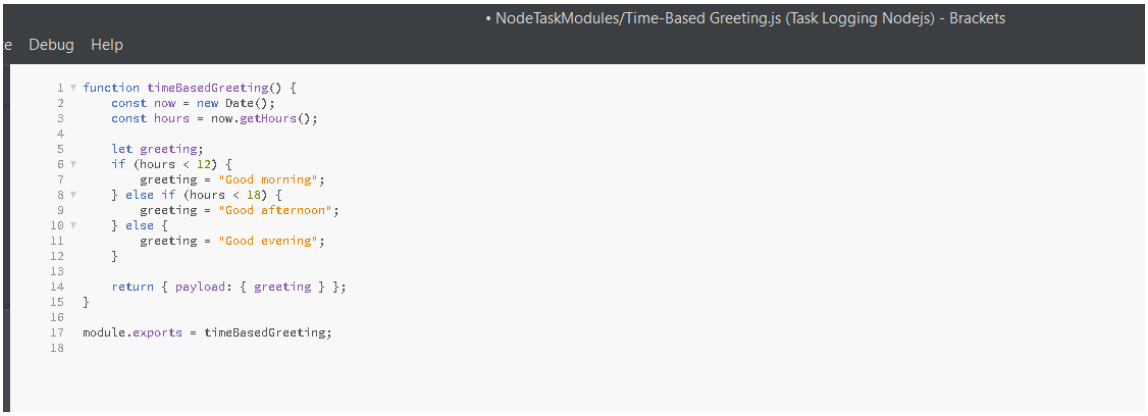
Output:



4. **Time-Based Greeting**

   - Create a function that returns a greeting based on the current time (e.g., "Good morning", "Good afternoon").

Code:

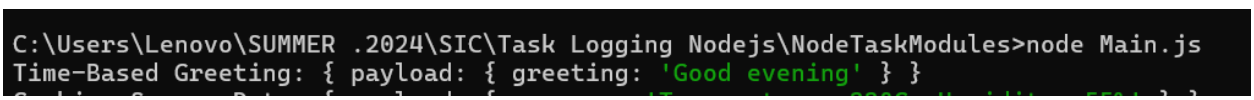


How?

The function returns a suitable greeting based on the time of day where time is classified into three periods: morning, afternoon, and evening.
var now = new Date();: Constructs a new Date object with the current time and date as its parameters.
Const hours are currently.Get the current hour from the Date object using the getHours() function. This number is an integer with a range of 0 to 23.

Choosing the Salutation:
1-Morning Salutation: "Good morning" is displayed if the hour is less than 12 (hours < 12).
2-Afternoon Greeting: "Good afternoon" is displayed if the hour is 12 or more but less than 18 (hours < 18).
3- Evening Salutation: "Good evening" is the greeting that appears if the hour is eighteen or greater.

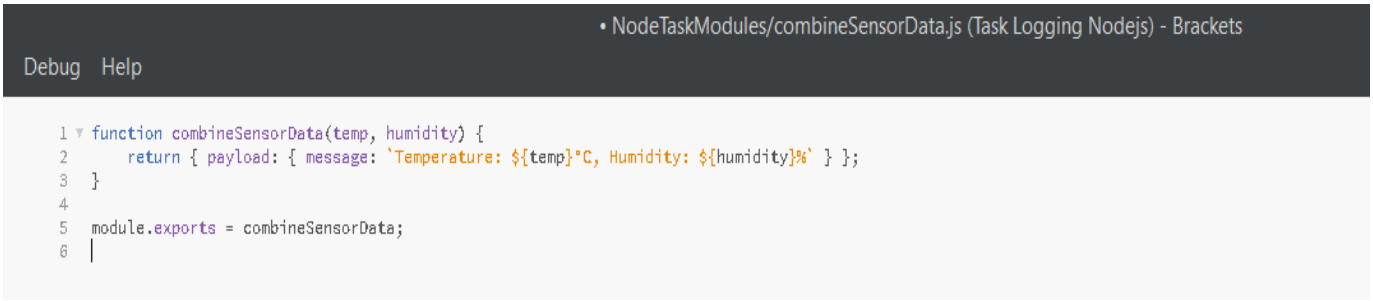
Output:



5. **Combine Sensor Data**

   - Write a function that combines data from two sensors (e.g., temperature and humidity) into one message.

Code:



How?

The approach eliminates sensor measurements that are less than a specified minimum level with only those measuring equal or greater than the same surviving it.

In this sense filtering ensures that only useful and significant sensor data remains which is applicable to cases where readings above certain values are important or need to be taken into account.

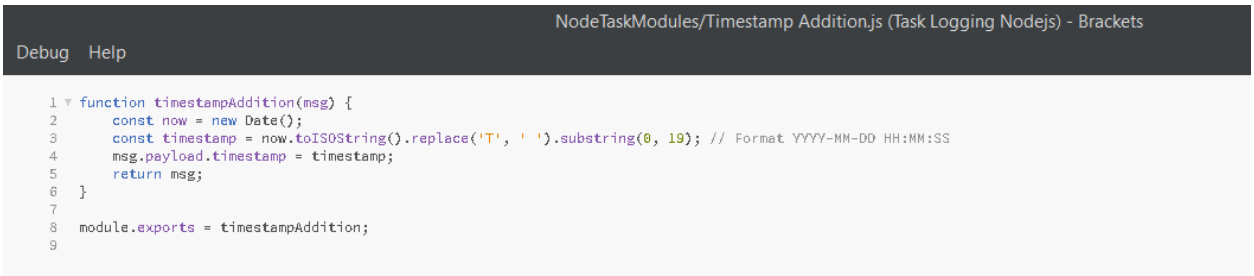
Data endorsement: This procedure helps maintain data quality and relevance by removing values below set threshold points

Output:

6. **Timestamp Addition**

   - Write a function that adds a timestamp to the message payload. Include the current date and time in the format `YYYY-MM-DD HH:MM:SS`.

Code:



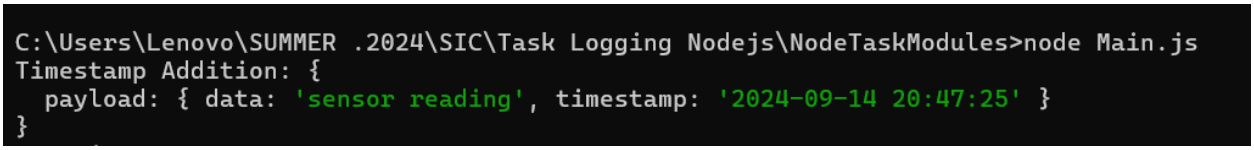
How?

This function adds a timestamp to a message object. It uses new Date() to generate the current date and time, converts that into common format of YYYY-MM-DD HH:MM:SS by converting the date to ISO format, replacing “T” with a space and removing milliseconds and time zone details.

The formatted timestamp is added then as new property timestamp within the payload object of message. Now this enriched message contains timestamp and is returned.

Thus one can keep track of when the message was processed or created using this function.

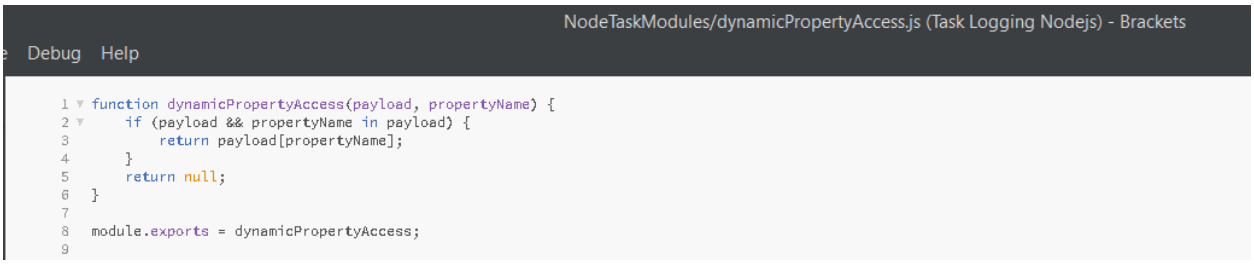
Output:



7. **Dynamic Property Access**

- Write a function that accesses a property of `msg.payload` based on a property name provided in another input (e.g., `msg.propertyName`).

Code:



How?

This function dynamically retrieves the value of a property from an object based on a property name.
The dynamicPropertyAccess function takes an object and a property name as inputs, accessing the value of that property based on its name.

Where,
payload: It is the object we want to get property from.
propertyName: This will be property whose value we want to retrieve from payload object.

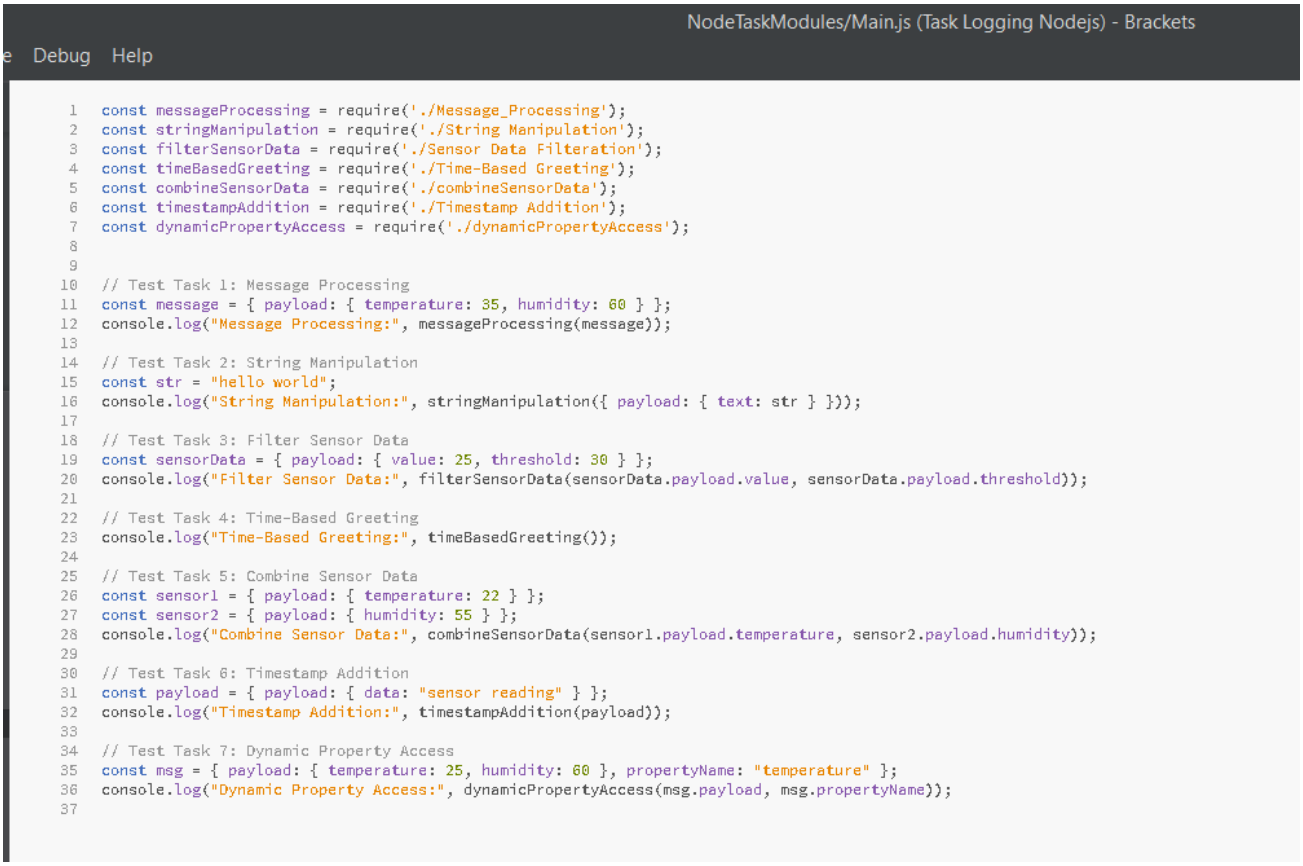
Property Access Logic:

Validation: Firstly, the function checks if payload is a sensible thing, meaning it should not be null or undefined then does propertyName exist within the keys for this particular object (propertyName in payload).
If both conditions are true, it returns value corresponding to propertyName within given payload (payload[propertyName]).
Alternatively if either thing does not hold or if propertyName does not exist among some set of keys in respective payloads then output would be equivalent to no value whatsoever.

Output:

```
C:\Users\Lenovo\SUMMER .2024\SIC\Task Logging Nodejs\NodeTaskModules>node Main.js
Dynamic Property Access: 25
```

Main.js:




```
const messageProcessing = require('./Message_Processing');
const stringManipulation = require('./String Manipulation');
const filterSensorData = require('./Sensor Data Filteration');
const timeBasedGreeting = require('./Time-Based Greeting');
const combineSensorData = require('./combineSensorData');
const timestampAddition = require('./Timestamp Addition');
const dynamicPropertyAccess = require('./dynamicPropertyAccess');


// Test Task 1: Message Processing
const message = { payload: { temperature: 35, humidity: 60 } };
console.log("Message Processing:", messageProcessing(message));

// Test Task 2: String Manipulation
const str = "hello world";
console.log("String Manipulation:", stringManipulation({ payload: { text: str } }));

// Test Task 3: Filter Sensor Data
const sensorData = { payload: { value: 25, threshold: 30 } };
console.log("Filter Sensor Data:", filterSensorData(sensorData.payload.value, sensorData.payload.threshold));

// Test Task 4: Time-Based Greeting
console.log("Time-Based Greeting:", timeBasedGreeting());

// Test Task 5: Combine Sensor Data
const sensor1 = { payload: { temperature: 22 } };
const sensor2 = { payload: { humidity: 55 } };
console.log("Combine Sensor Data:", combineSensorData(sensor1.payload.temperature, sensor2.payload.humidity));

// Test Task 6: Timestamp Addition
const payload = { payload: { data: "sensor reading" } };
console.log("Timestamp Addition:", timestampAddition(payload));

// Test Task 7: Dynamic Property Access
const msg = { payload: { temperature: 25, humidity: 60 }, propertyName: "temperature" };
console.log("Dynamic Property Access:", dynamicPropertyAccess(msg.payload, msg.propertyName));

```

Problems Faced:

1.Errors of Module Not Found

Problem: A regular mistake that happens when the directory of modules is wrong is resultant to the Error: Cannot find module.

```
C:\Users\Lenovo\SUMMER .2024\SIC\Task Logging
Nodejs\NodeTaskModules>node Main.js
node:internal/modules/cjs/loader:1251
  throw err;
  ^

Error: Cannot find module 'messageProcessing'
```

Cause: This happens when the require() statement doesn't match with a file name exactly or has changed its arrangement.

- ✓ **Solution:** To Make sure to provide correct path to a required module.

2. Problems with Linting/ESLint

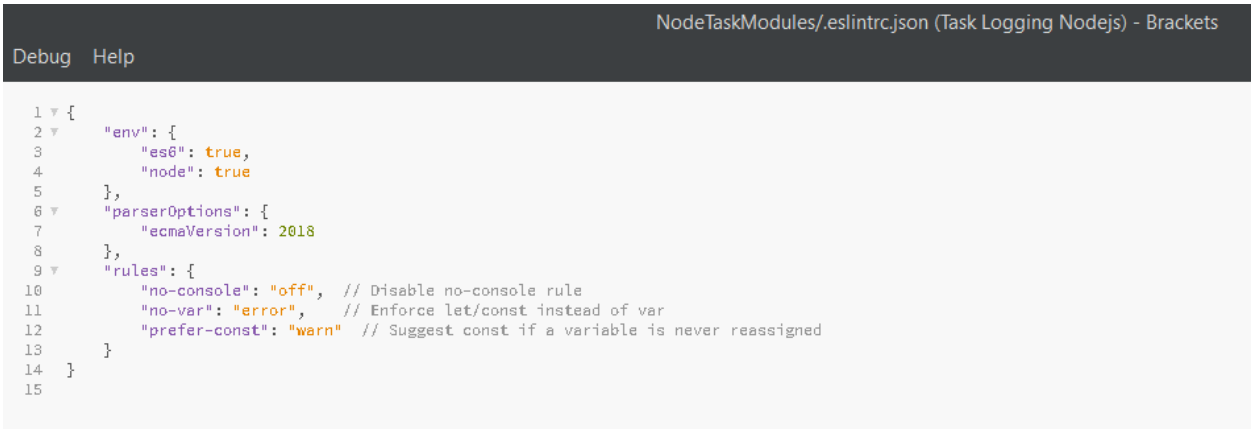
Problem: If we don't abide by contemporary JavaScript standards, one may encounter errors like Unexpected var, prefer-const or no-console.

```
ERROR: Unexpected console statement. [no-console]
console.log("String Manipulation:", stringManipulation(str));
20
ERROR: Unexpected console statement. [no-console] console.log("Filter
Sensor Data:", filterSensorData(25, 30));
23
```

Cause: Old scripts generally use var rather than let or const whereas production environments usually avoid console.log statements.

- ✓ **Solution:** update ESLint configurations so as to take care of these mistakes.

.eslintrc.json



3. For Timestamp, the Timing/Clock Issues

Problem: If the time zone or format is not right, it can result in wrong time values or inconsistent formatting when timestamps are added.

Cause: If not handled properly, JavaScript's Date() object may give unexpected formats.

- ✓ **Solution:**

To ensure that your chosen time format (e.g., YYYY-MM-DD HH:MM:SS) will be right use toISOString() with formatting adjustments.

4. Wrong msg Object

Problem: In messageProcessing function is expecting a specific format, but I passed an object with temperature and humidity directly, not inside a payload property.

```
C:\Users\Lenovo\SUMMER .2024\SIC\Task Logging
Nodejs\NodeTaskModules\Message_Processing.js:2
  const temp = msg.payload.temperature;
                           ^
```

- ✓ **Solution:** By ensuring messageProcessing function and the test data are aligned (with payload property handling) -> Handled Error successfully

```
// Test Task 1: Message Processing
const message = { payload: { temperature: 35, humidity: 60 } };
console.log("Message Processing:", messageProcessing(message));
```