# Module 2 Assignment: JavaScript language

Devin Fries

Department of Information Technology, Arizona State University

IFT 458: Middleware Programming & Database Security

Dr. Dinesh Sthapit

September 1, 2023

**Index.js Notes**: The Index.js code is demonstrating different ways to define functions, including regular functions and fat arrow functions, and how to call them to obtain specific results. Greeting Function Examples:

The code includes several examples of a greeting function that takes a studentName as a parameter and returns a greeting message.

There are three implementations: a regular function, a fat arrow function, and a more concise fat arrow function.

Full Name Function:

The code also demonstrates a function called fullName that takes three parameters: firstName, middleName, and lastName.

It returns a formatted full name with the last name, middle name, and first name in that order, separated by commas.

Function Calls:

The code provides an example of how to call the fullName function with specific values for firstName, middleName, and lastName to obtain a formatted full name.

It assigns the result to the titleName variable and then logs the result to the console. The code includes commented-out sections that demonstrate alternative ways of defining functions and calling them.

Variable Assignment and Logging:

The code assigns the result of the fullName function call to the titleName variable and logs the result to the console using console.log. I think this code serves as an educational example illustrating different ways to define functions in JavaScript, emphasizing the use of fat arrow functions for concise function definitions. It also demonstrates how to call these functions and assign their results to variables for further use.

**functionDemo1.js Notes**: The provided JavaScript code, named "functionDemo1.js," contains a series of variable declarations and function definitions.

Variable Declarations:

age is declared and initialized with the value 10, which is an integer.

salary is declared and initialized with the value 15.00, which is a floating-point number (double).

There are commented-out lines attempting to log the variable types, which can be helpful for debugging.

Functions:

displayGreetings is defined as an anonymous function that takes two parameters, name and year, and logs a greeting message to the console. The message includes the provided name and year.

displayGreetingsWithEmoji is another anonymous function with the same parameters, but it adds emojis to the greeting message.

greet is a higher-order function that takes three parameters: name, year, and func. It invokes the provided func with the given name and year.

Function Calls:

The greet function is called twice with different names and years. In the first call, it uses the displayGreetingsWithEmoji function to display a new year's greeting with emojis, while in the second call, it uses the displayGreetings function for a standard greeting. The code showcases the use of variables, including integers and floating-point numbers. It demonstrates the concept of functions, both regular functions (displayGreetings, displayGreetingsWithEmoji) and higher-order functions (greet) that accept other functions as parameters.

The code uses template literals (backticks) to create formatted strings, which allow the inclusion of variables and expressions within strings using ${}. It provides a simple example of function callback by passing different greeting functions to the greet function, showcasing how you can customize behavior by passing functions as arguments.

There's commented-out code that shows the intention to log variable types, which can be useful for debugging but is currently inactive. I would say this code serves as a basic demonstration of JavaScript variable types, functions, and the concept of higher-order functions and callbacks.

**Objects.js Notes**:This code works with arrays and objects to store and manipulate data related to students and their grades.

Array Declaration:

Two arrays are defined: students and grades.

The students array contains the names of five students.

The grades array holds numerical grades corresponding to these students.

Array Access:

There are several console.log statements that were used for accessing elements within the arrays.

For example, students[0] would access the first student's name ('John'), and grades[2] would access the third grade in the grades array (89).

students.length and grades.length are used to determine the number of elements in each array.

Array Element Access by Index:

Some console.log statements are commented out, which would have displayed specific elements in the arrays, such as the first student or the third grade.

The notation [index] is used to access elements in an array, where index is the position of the element (starting from 0).

Student Information as Objects:

An array named studentInfo is declared.

This array contains objects, each representing a student's information.

Each object has two properties: 'name' for the student's name and 'grade' for the student's grade.

There are five objects in the studentInfo array, each representing a student's name and grade.

This code sets up data structures to store information about students and their grades. It uses arrays to hold names and grades and organizes more detailed student information in an array of objects. The code provides a foundation for further manipulation or analysis of this data, such as calculating averages, sorting, or performing other operations on the student information.

**Student.js Notes**: The provided student.js code defines an array called students and creates two student objects using JavaScript object literals.

Array Definition: The var students = []; line initializes an empty array called students where student objects will be stored.

Student Object (student): The first student object is defined with the following properties:

name: 'Jane Smith'

birthyear: 2002

course: 'IFT 458'

grade: 90

active: true

age (a function): This function calculates the age of the student by subtracting the birth year from the current year (assumed to be 2022).

Student Object (student2): The second student object is defined similarly to the first one, with the following differences:

name: 'Andy Moore'

birthyear: 2000

active: false

The age function now includes a conditional statement. If the student is active (active is true), it calculates the age; otherwise, it returns 0.

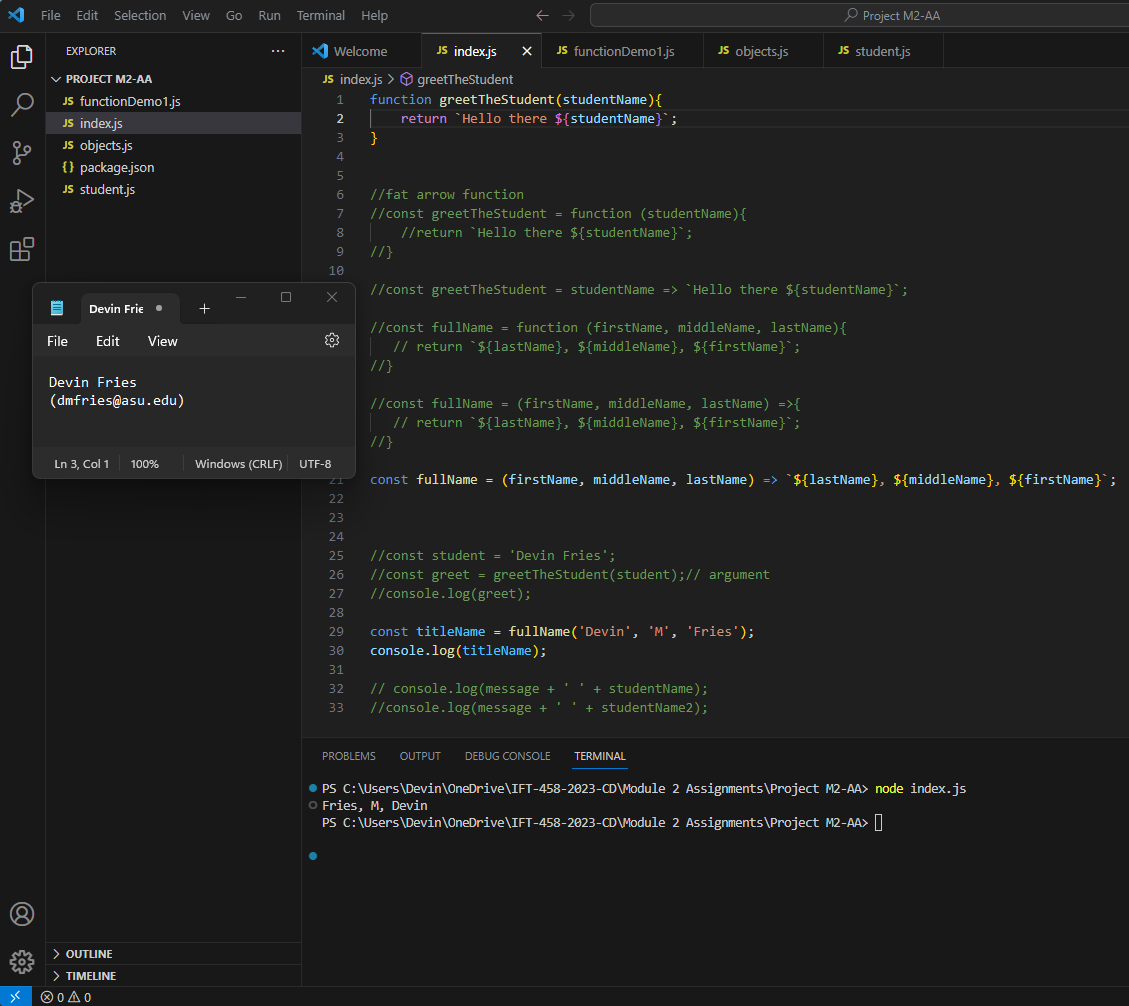
Pushing Students to the Array: Both student objects (student and student2) are pushed into the students array using students.push(student); and students.push(student2);.

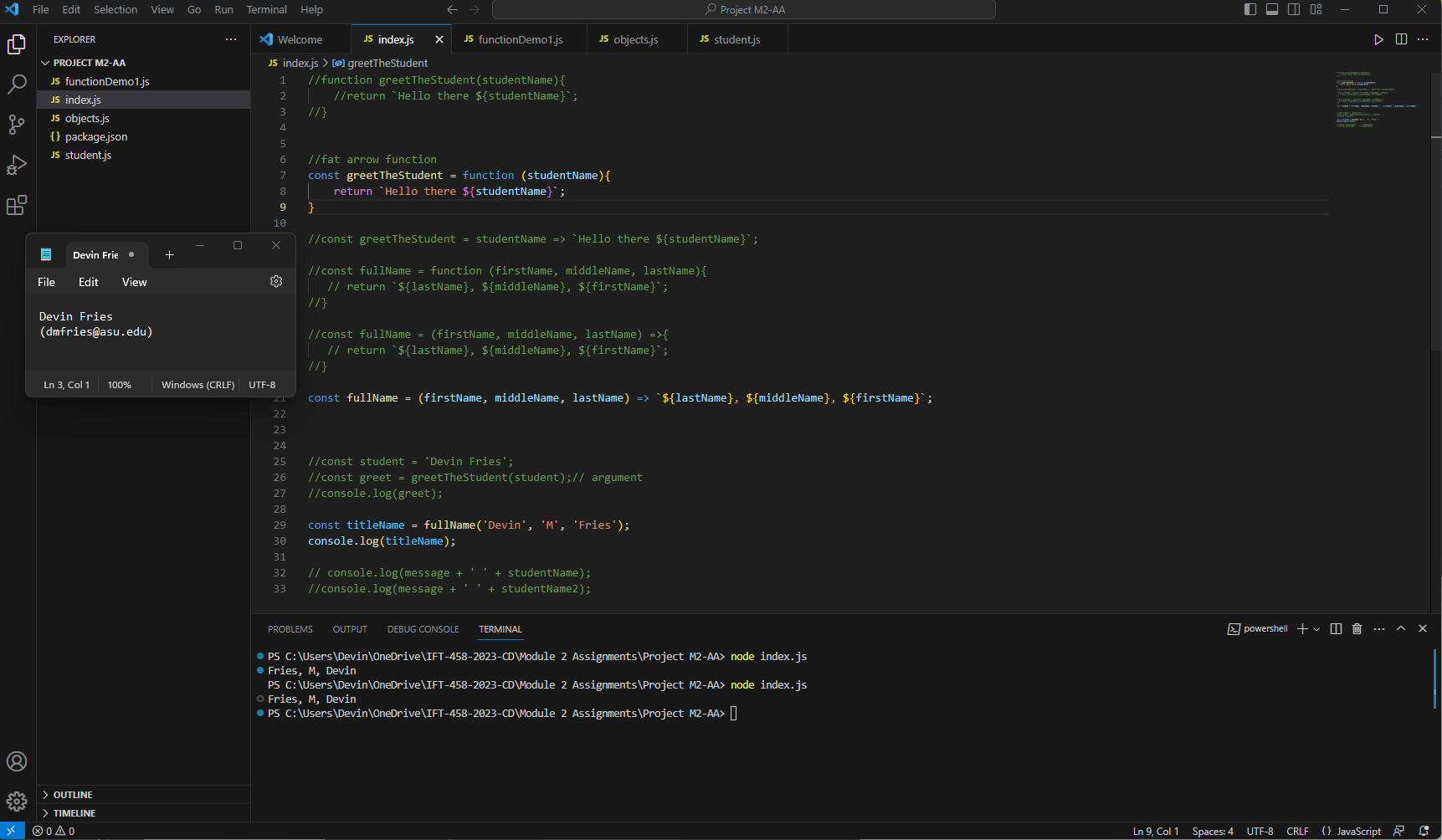
Console Logging: The code includes several console.log statements to display information:

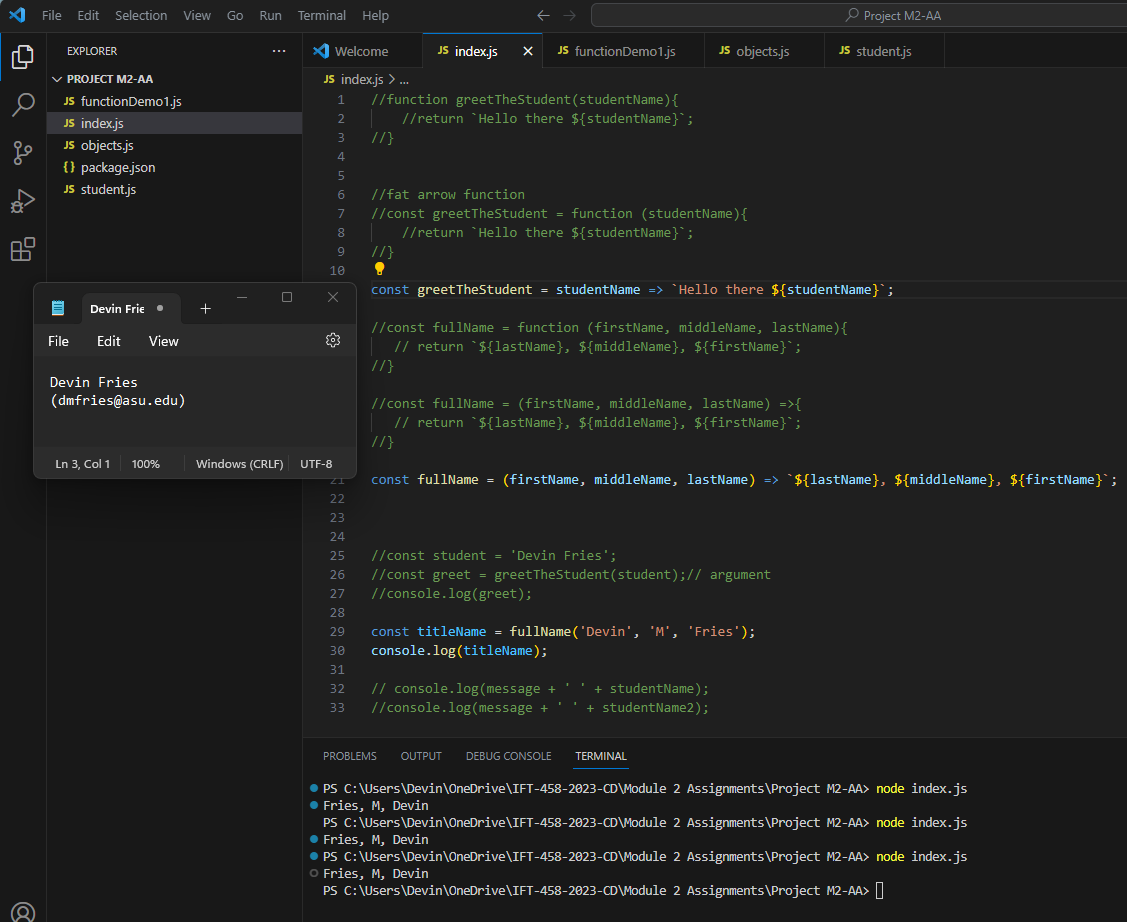
console.log(students); prints the entire students array, which contains both student objects.

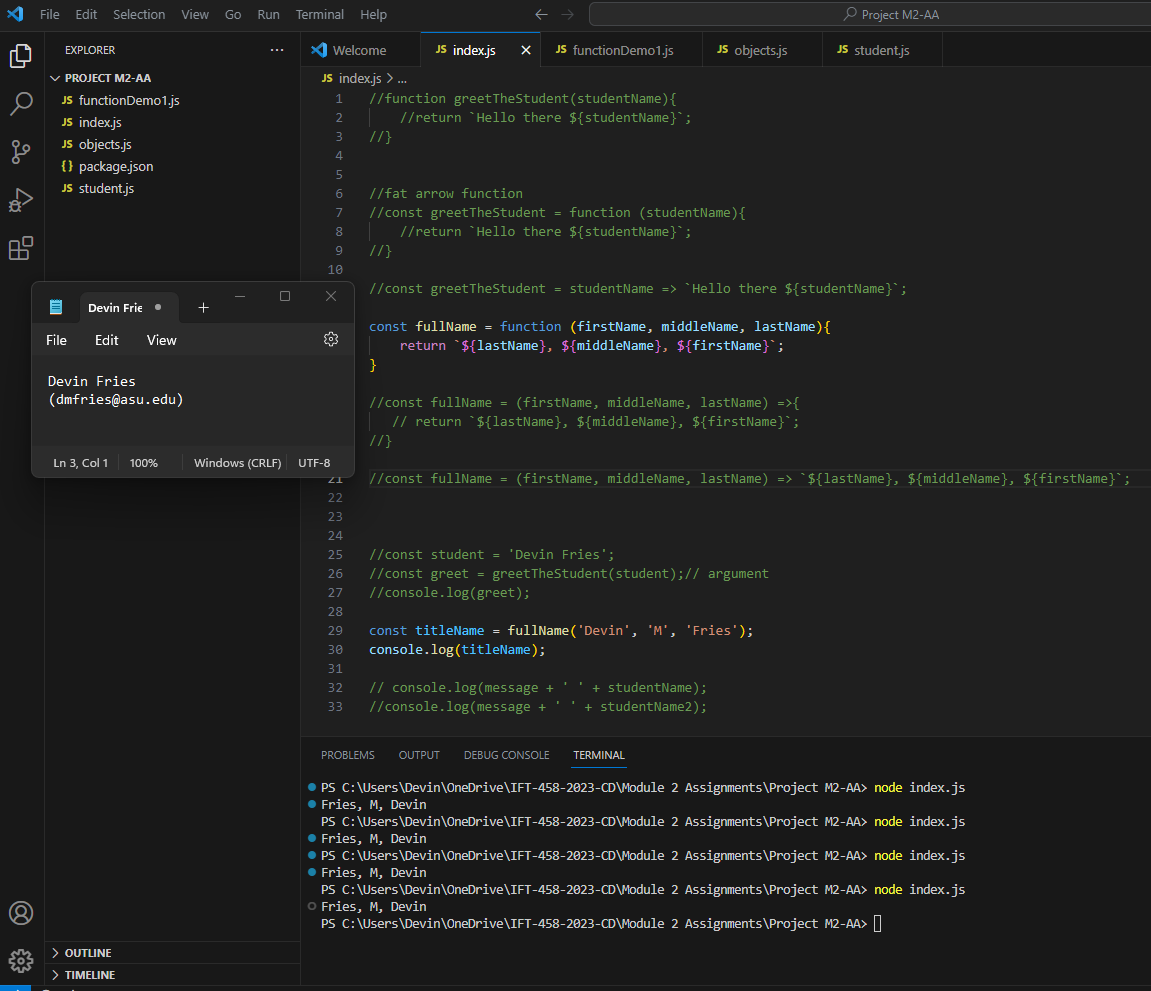
There are console.log statements that demonstrate different ways to access properties of a single student object (student). students.forEach((item)=> console.log(item.age())); iterates through each student in the array and prints their age by calling the age function for each student. This code defines two student objects, stores them in an array, and demonstrates how to access and manipulate the student data. It also showcases how you can use functions within objects to calculate and return values based on the object's properties.

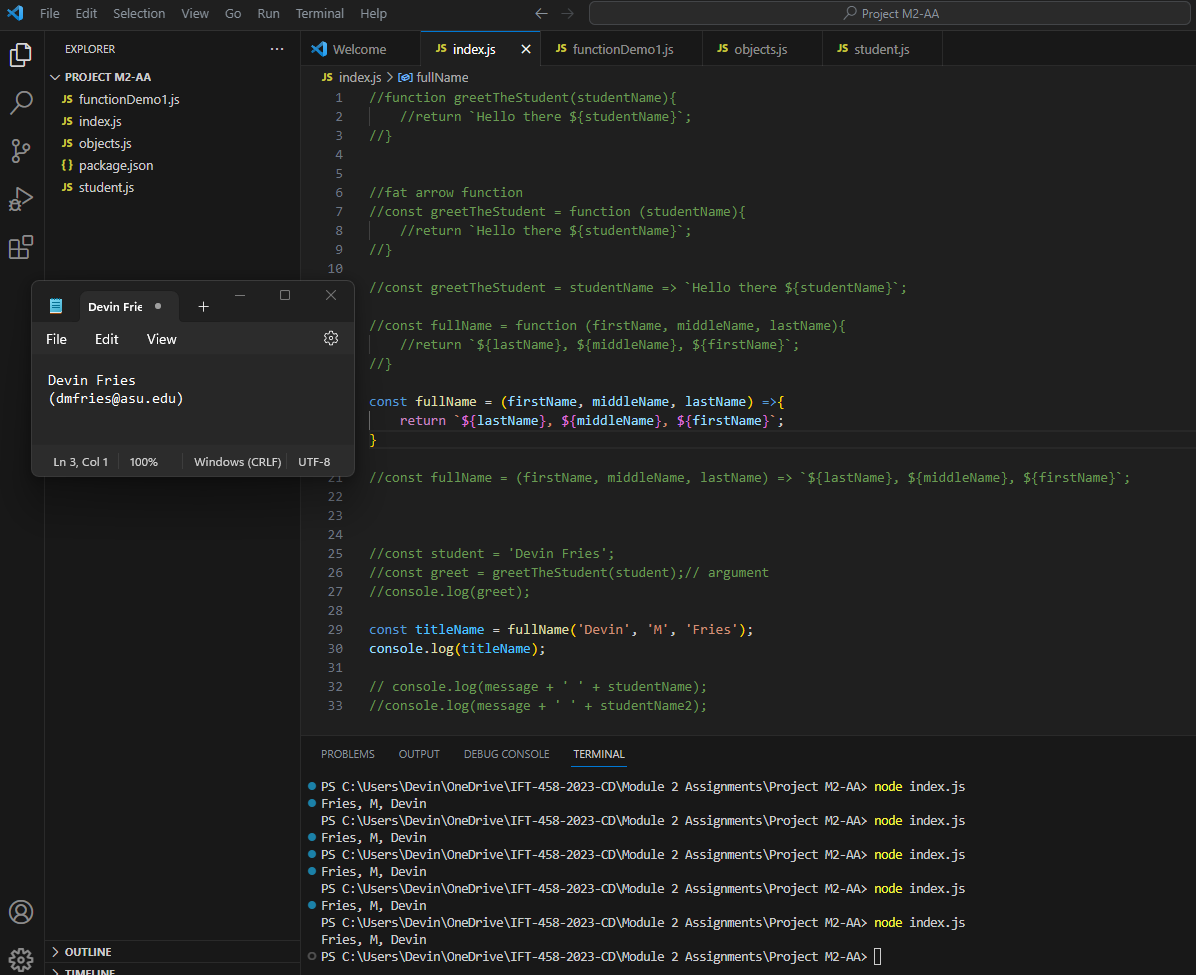
Screenshots:

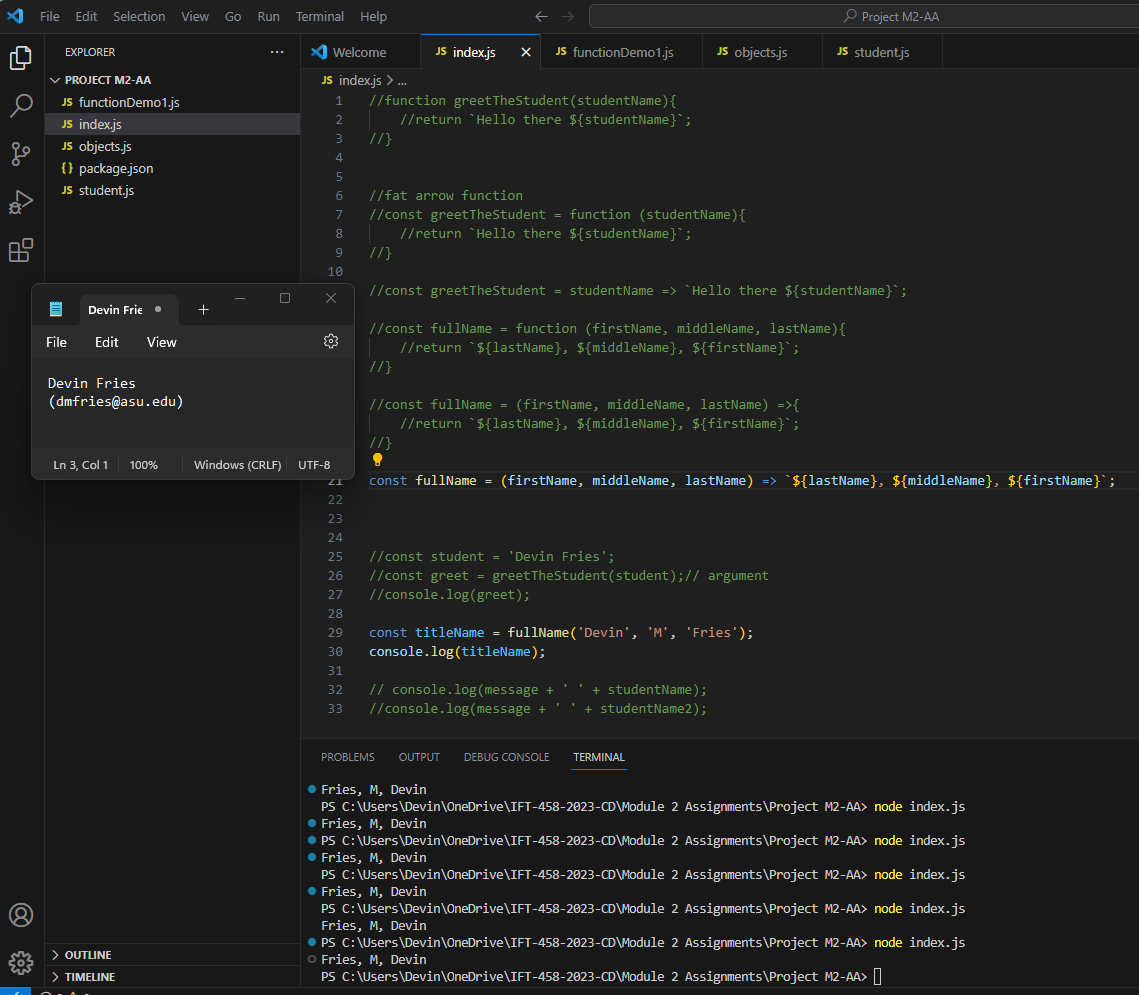
**Index.js:**

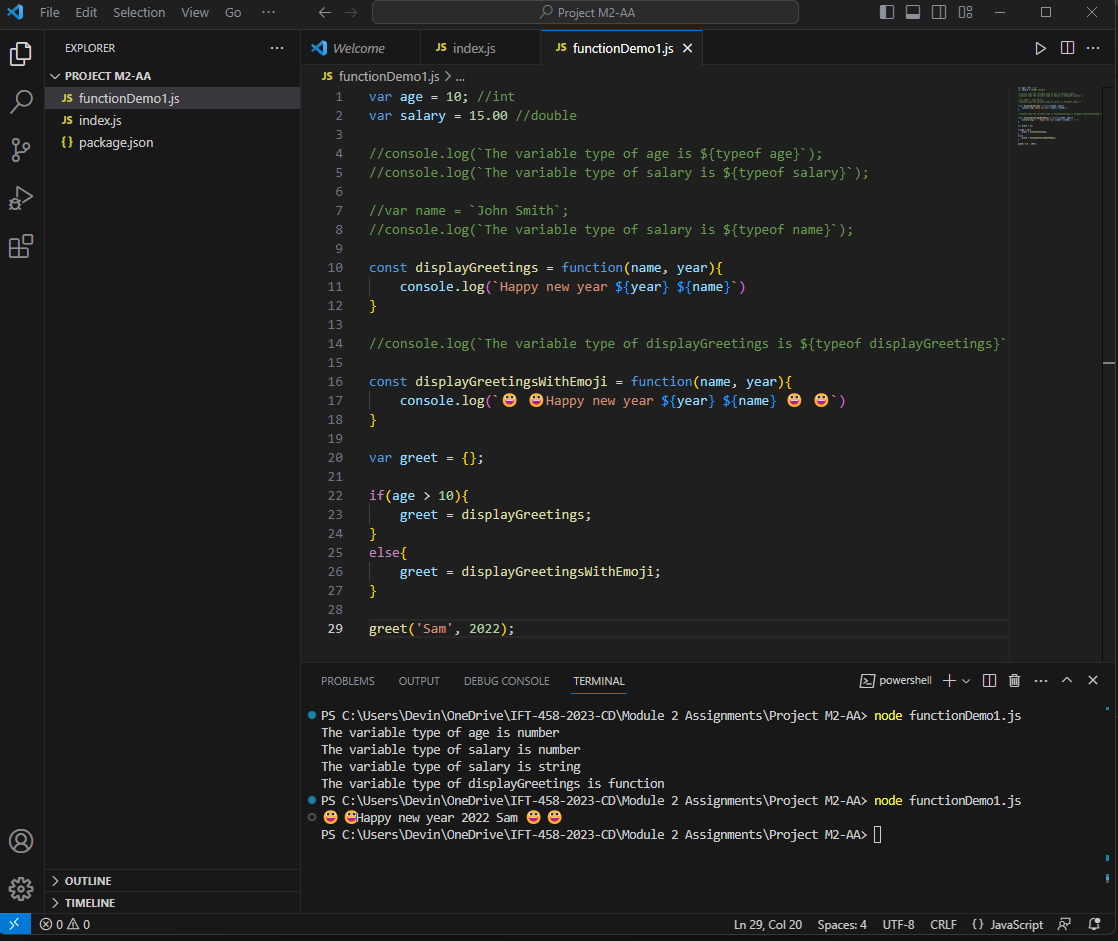


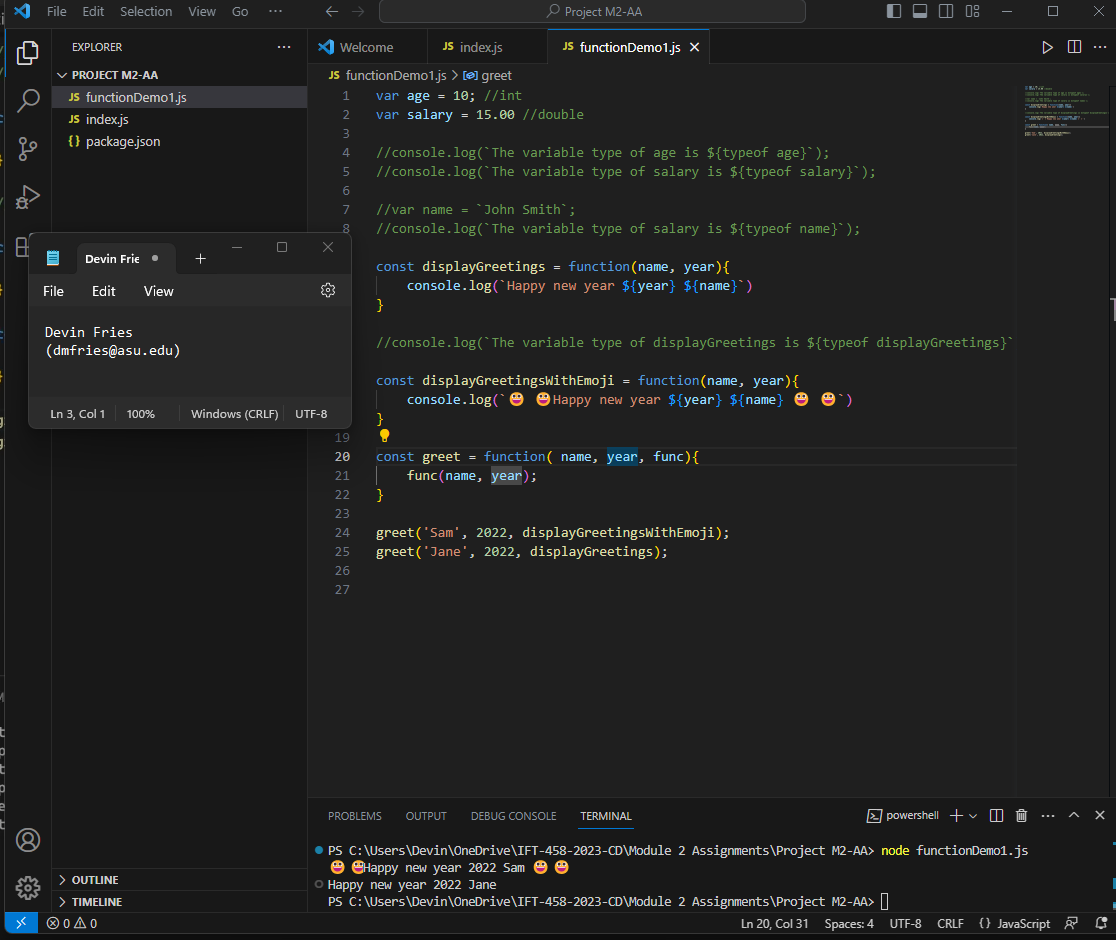


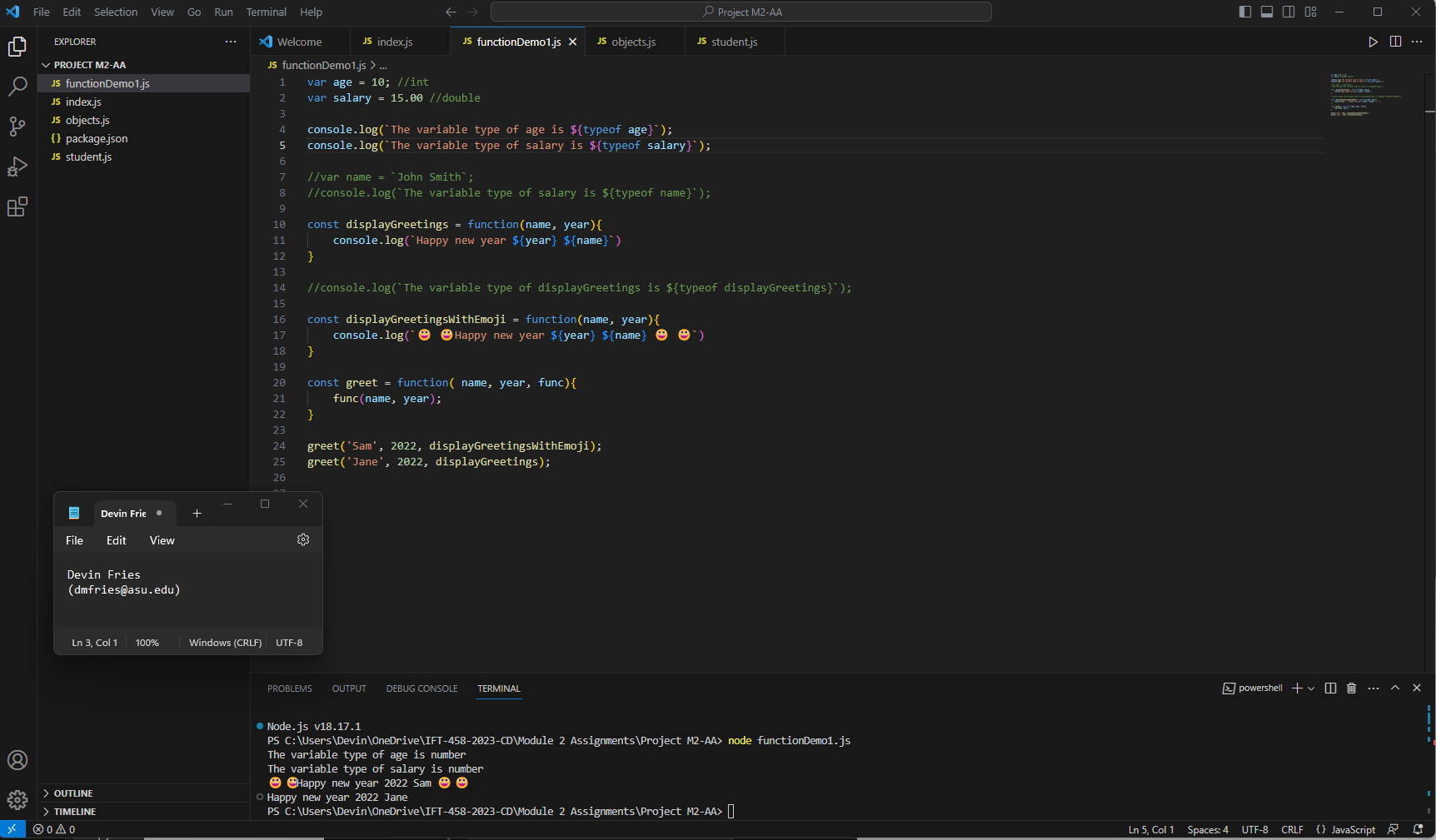


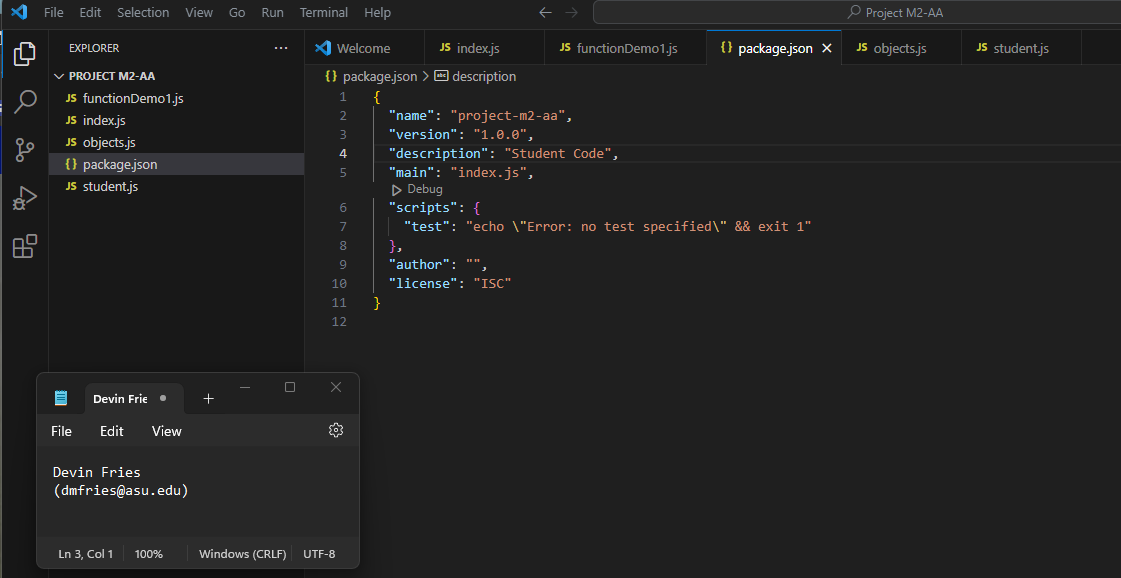


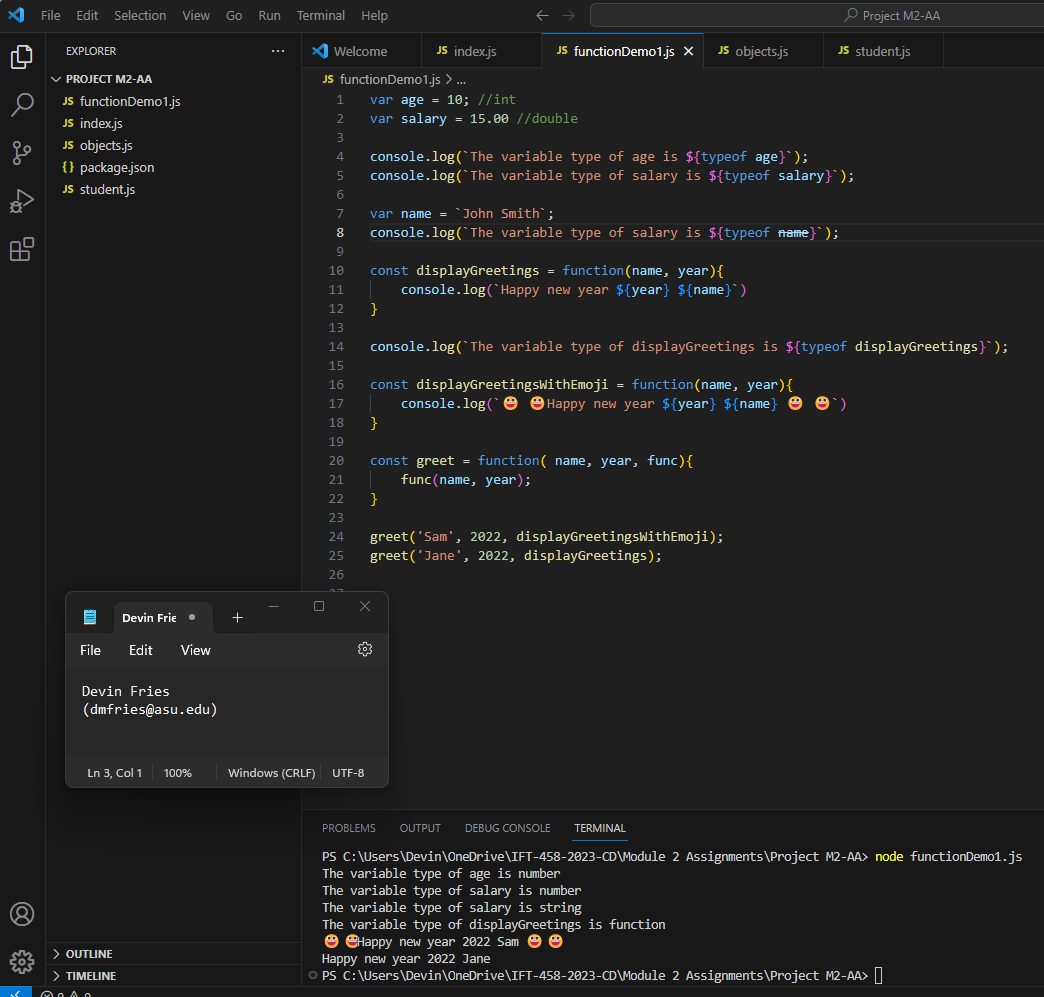


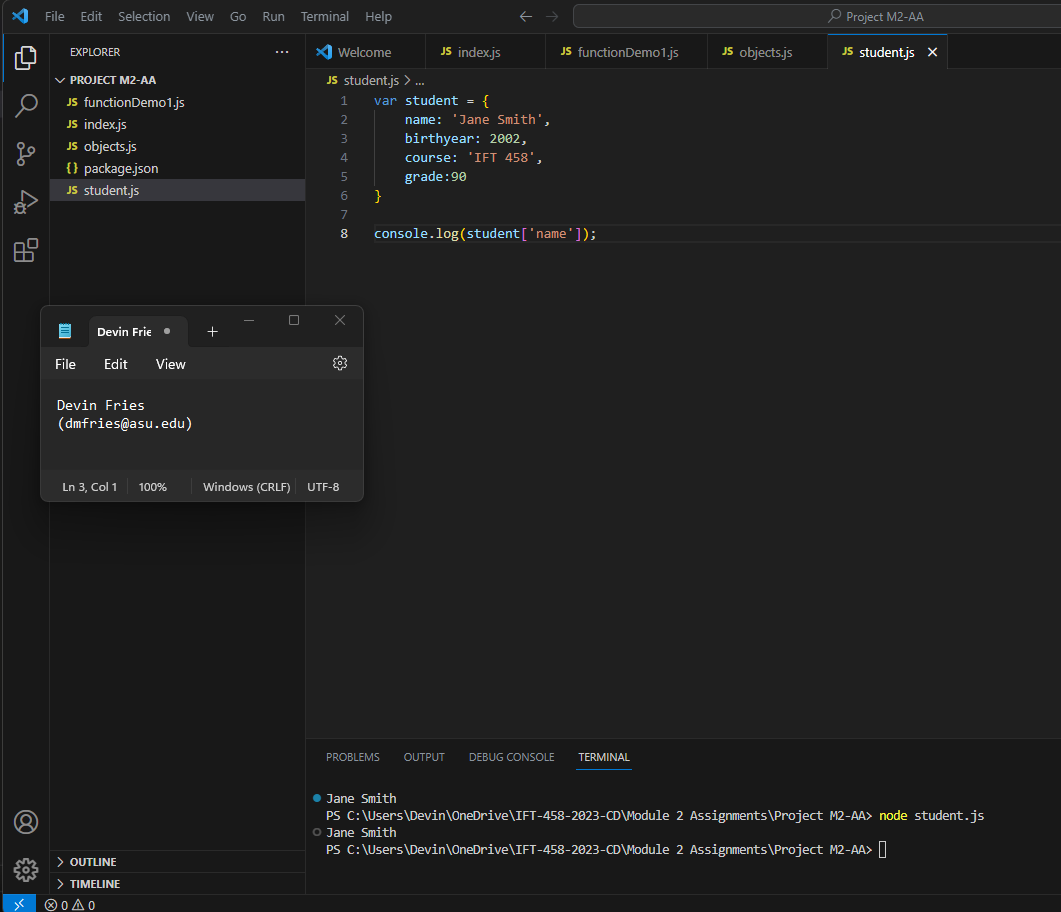
**functionDemo1,js: **

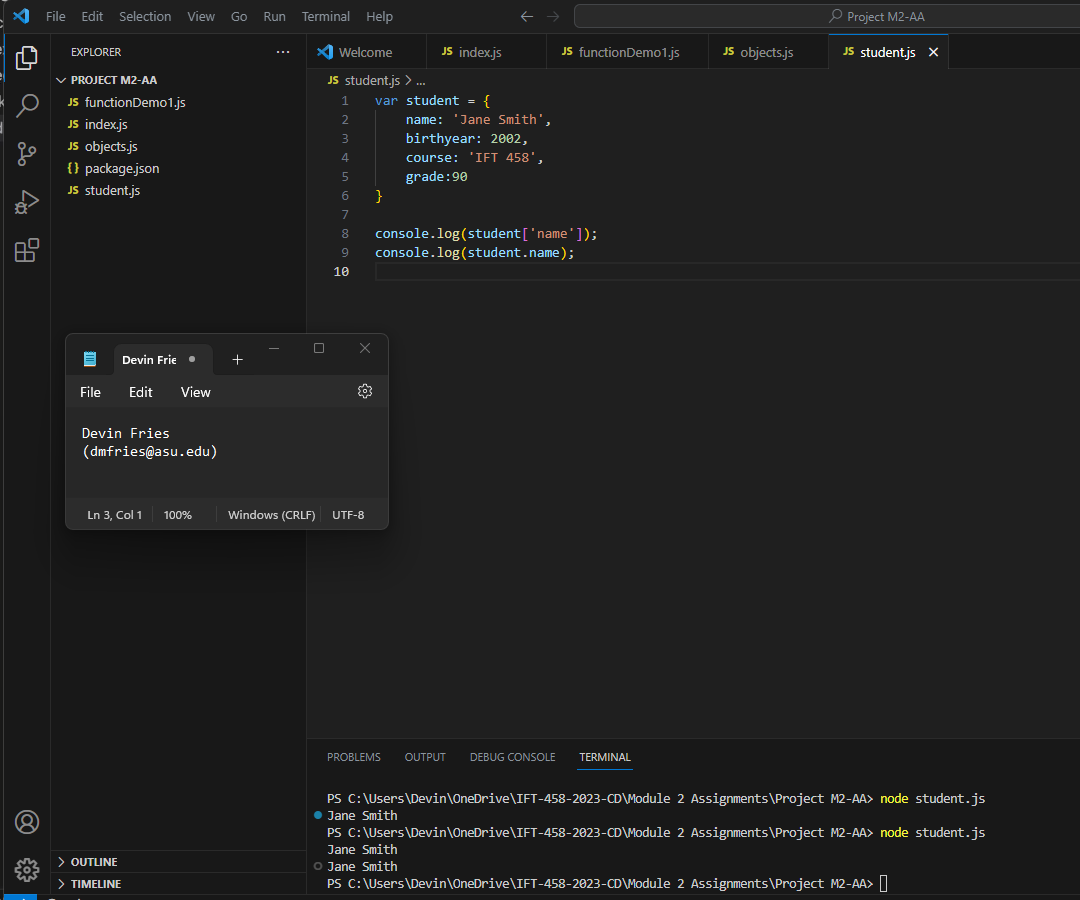
****

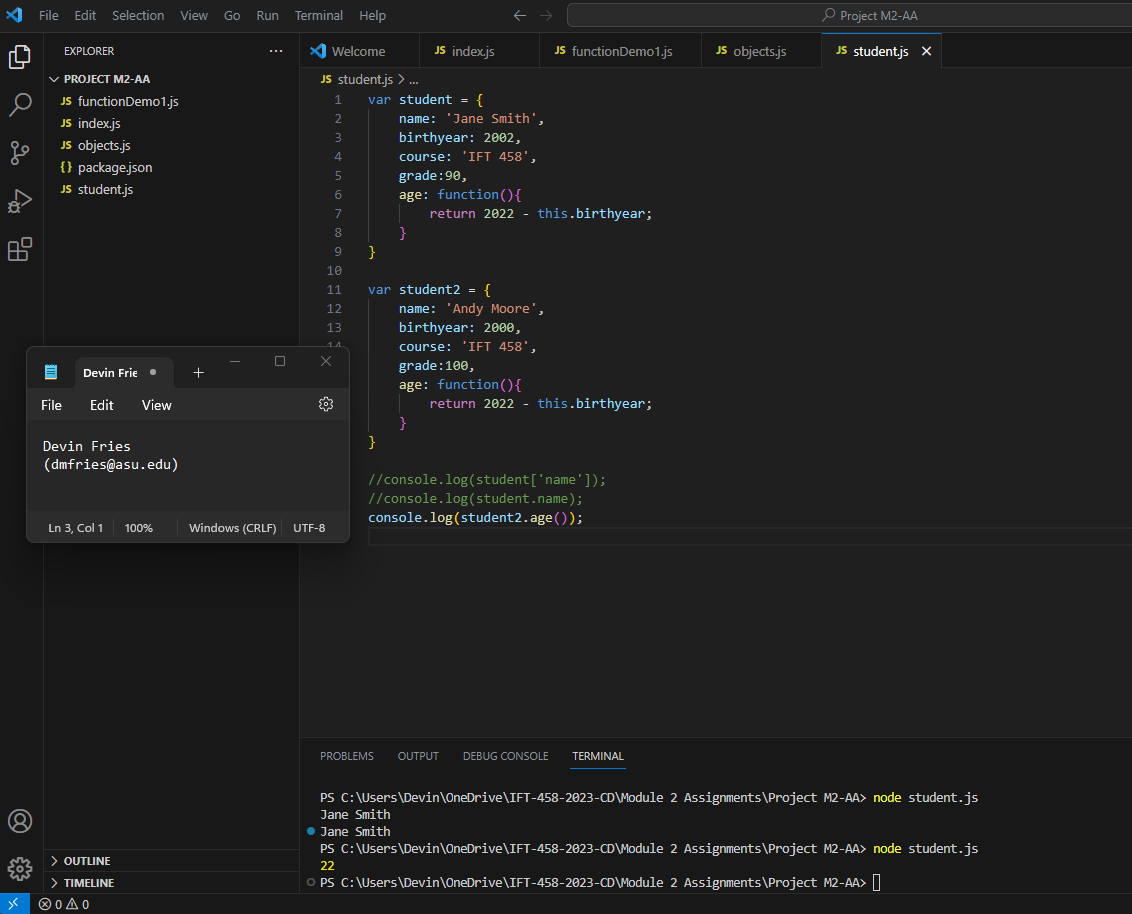
****

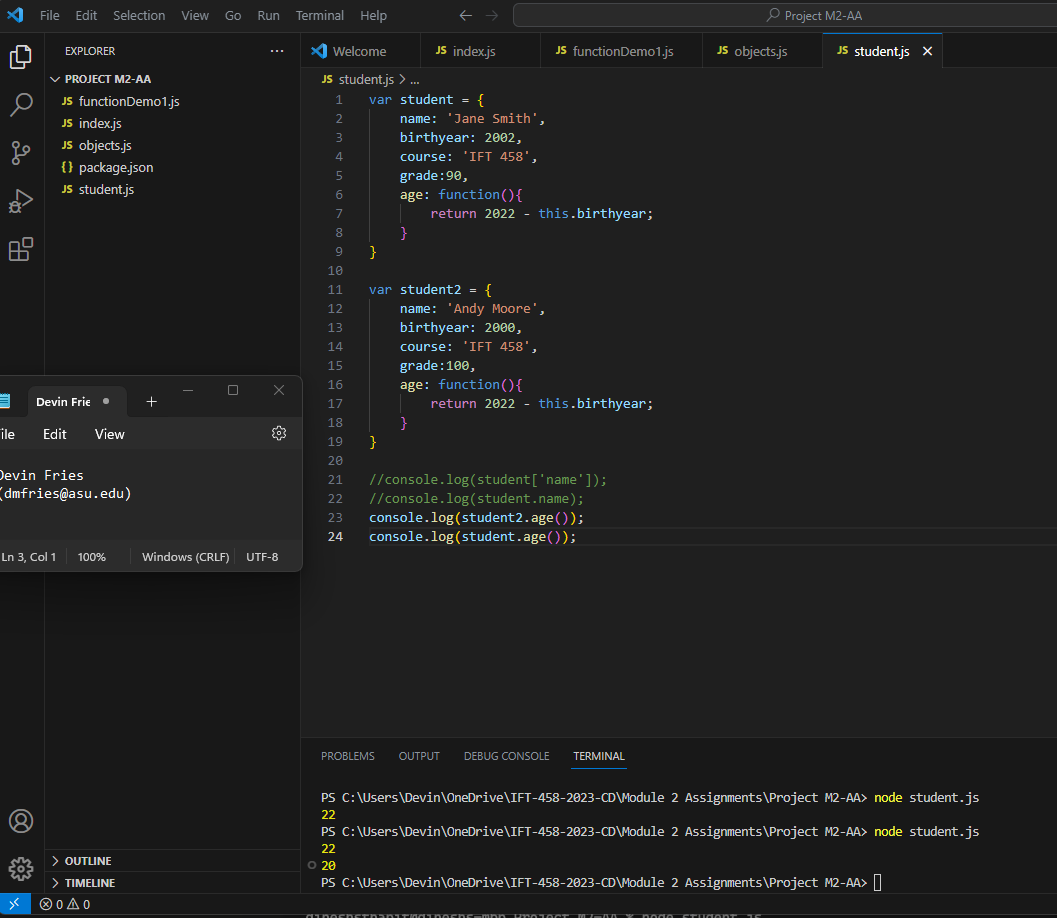
**JSON Student Data File:**

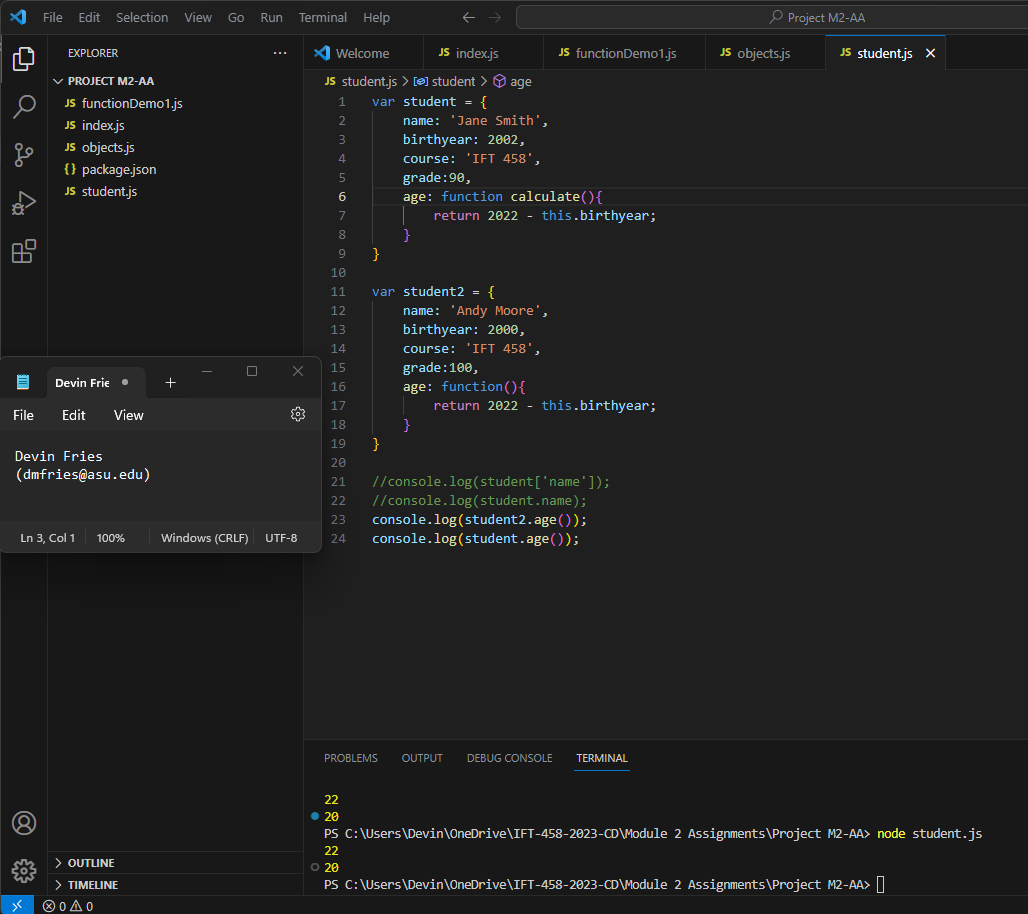
****

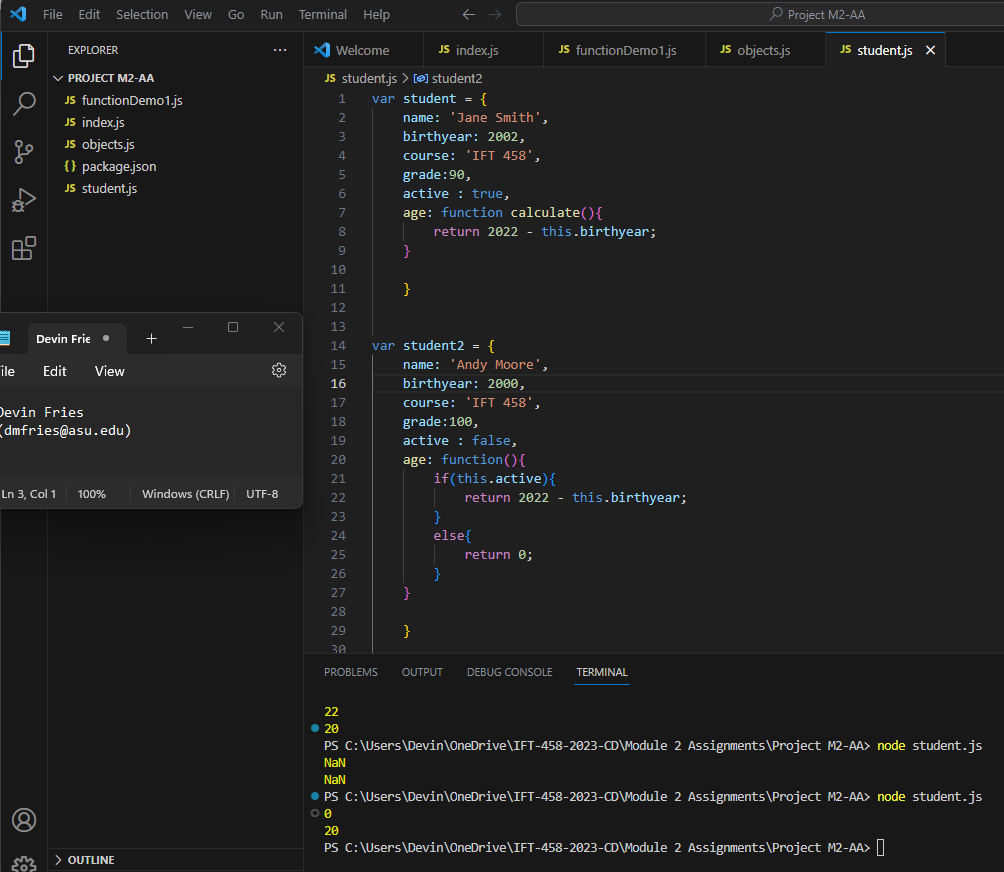
**Students.js: **

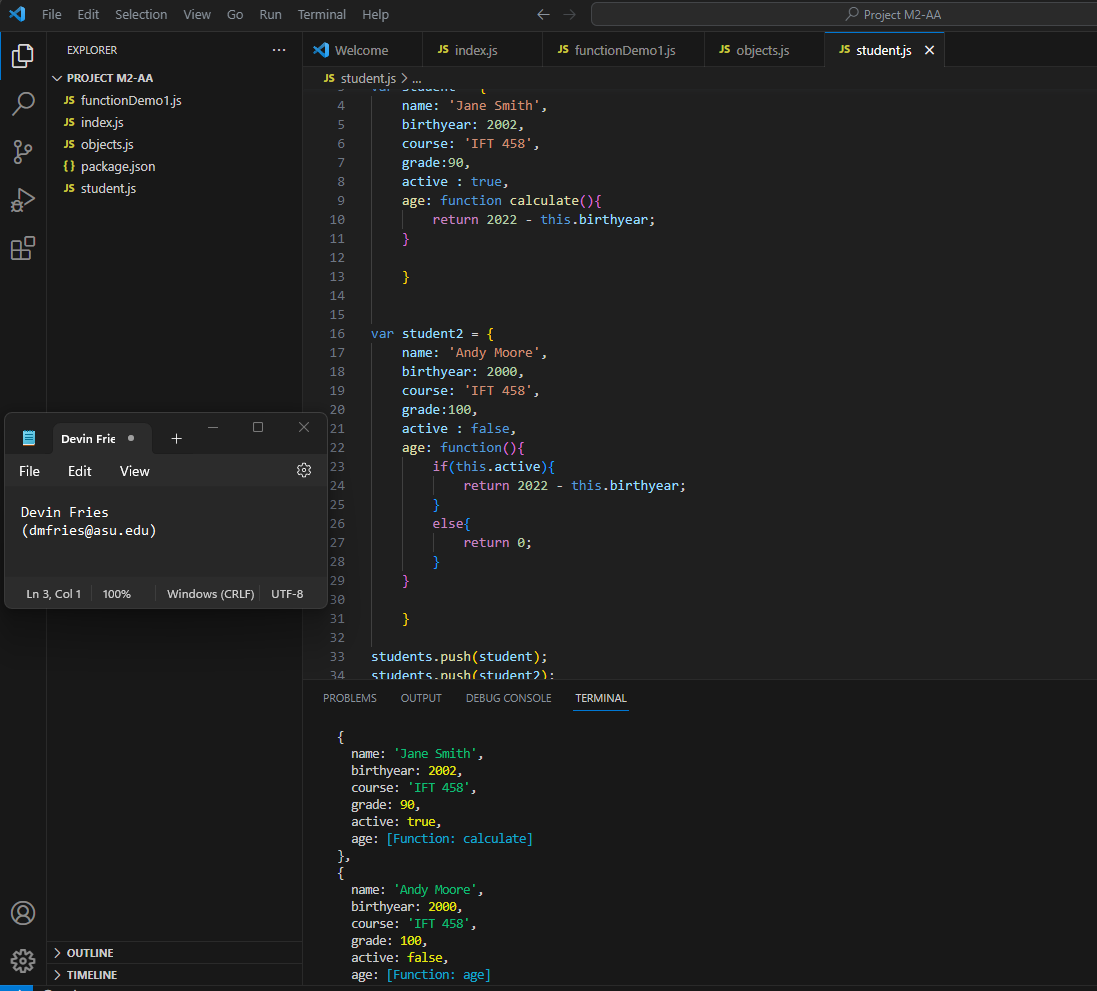
****

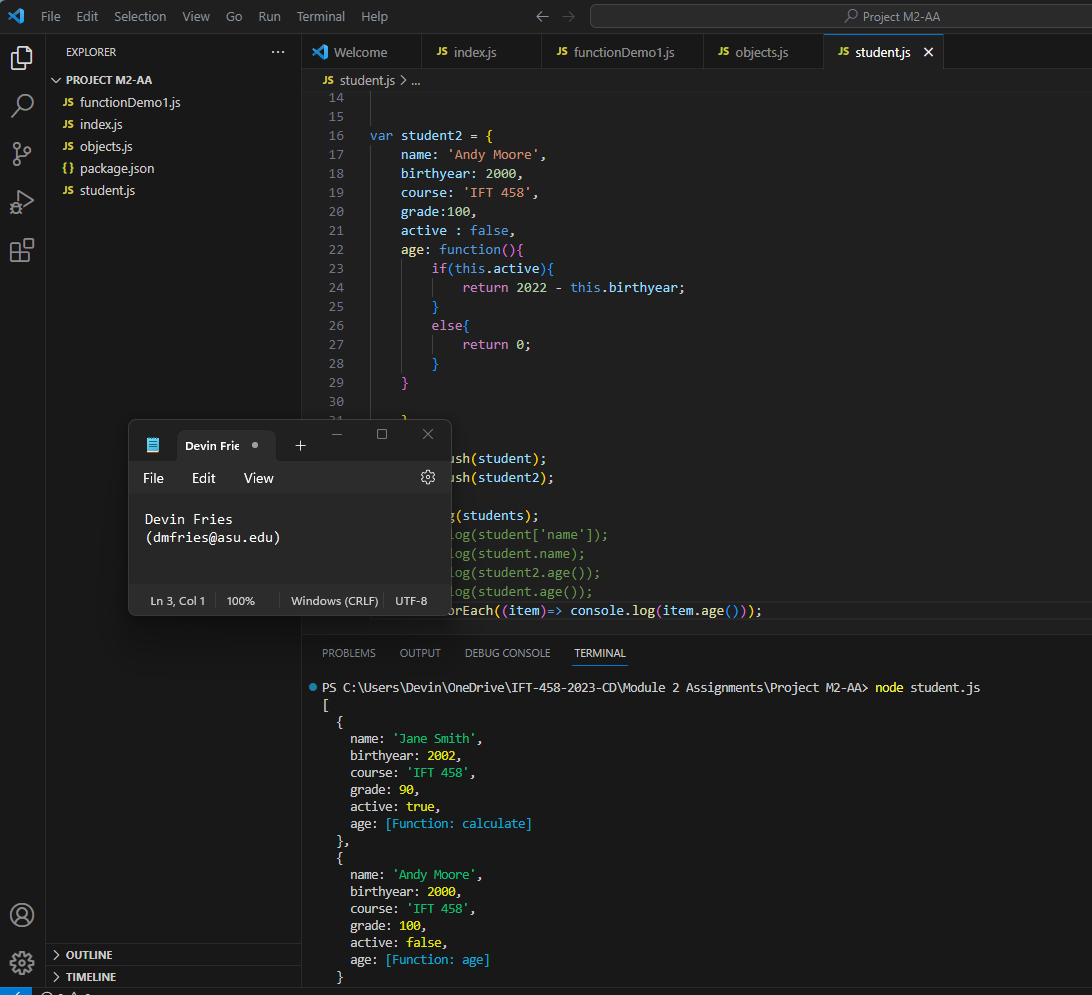
****

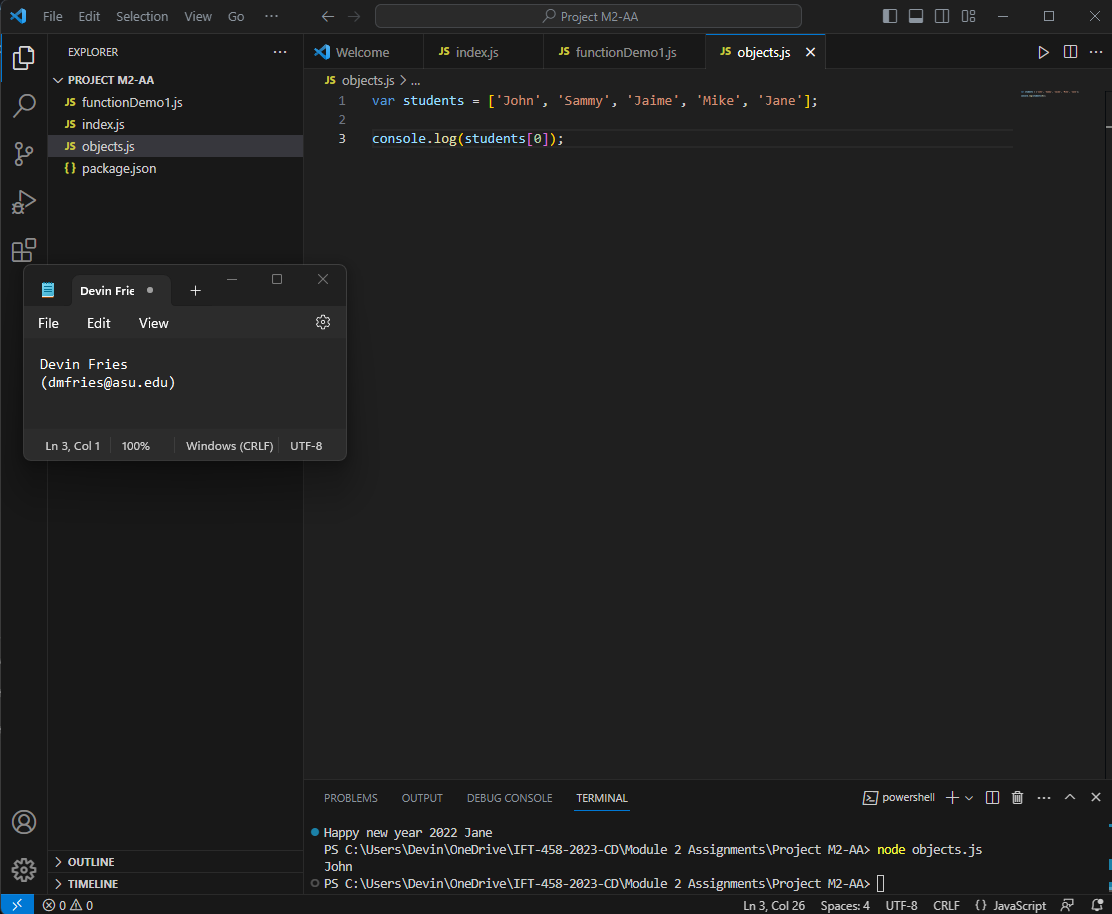
****

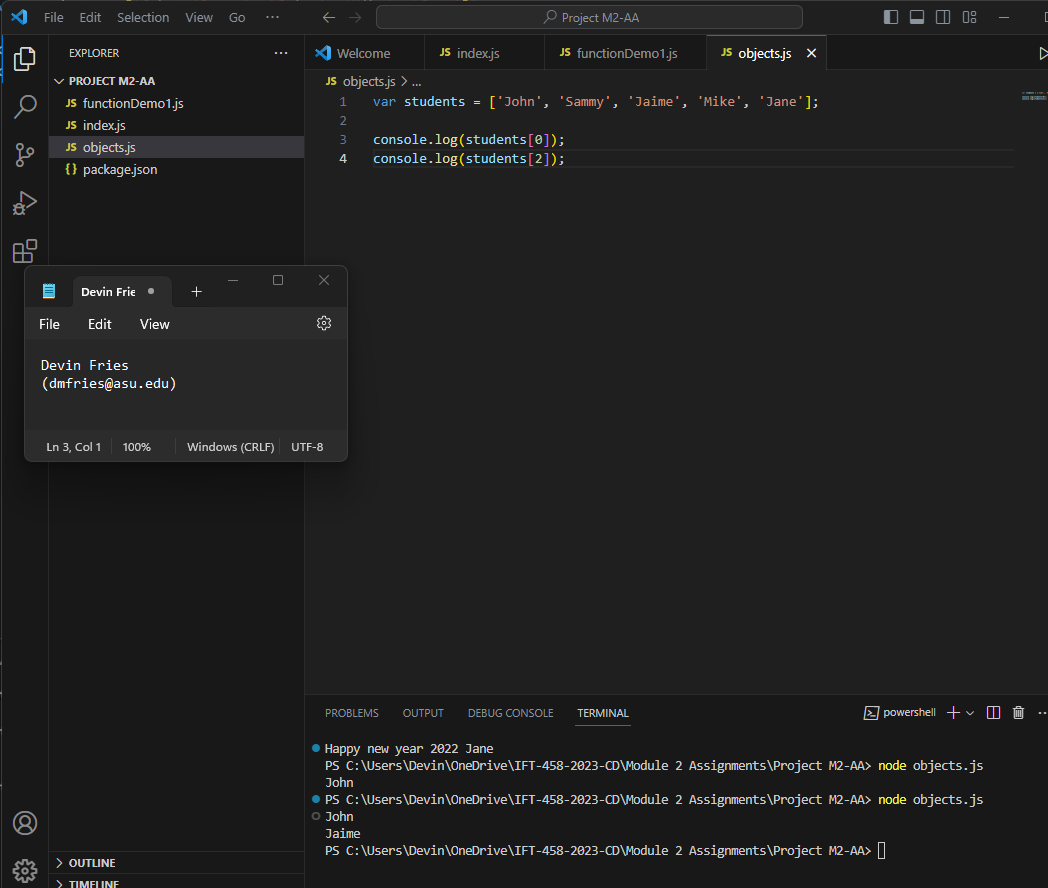
****

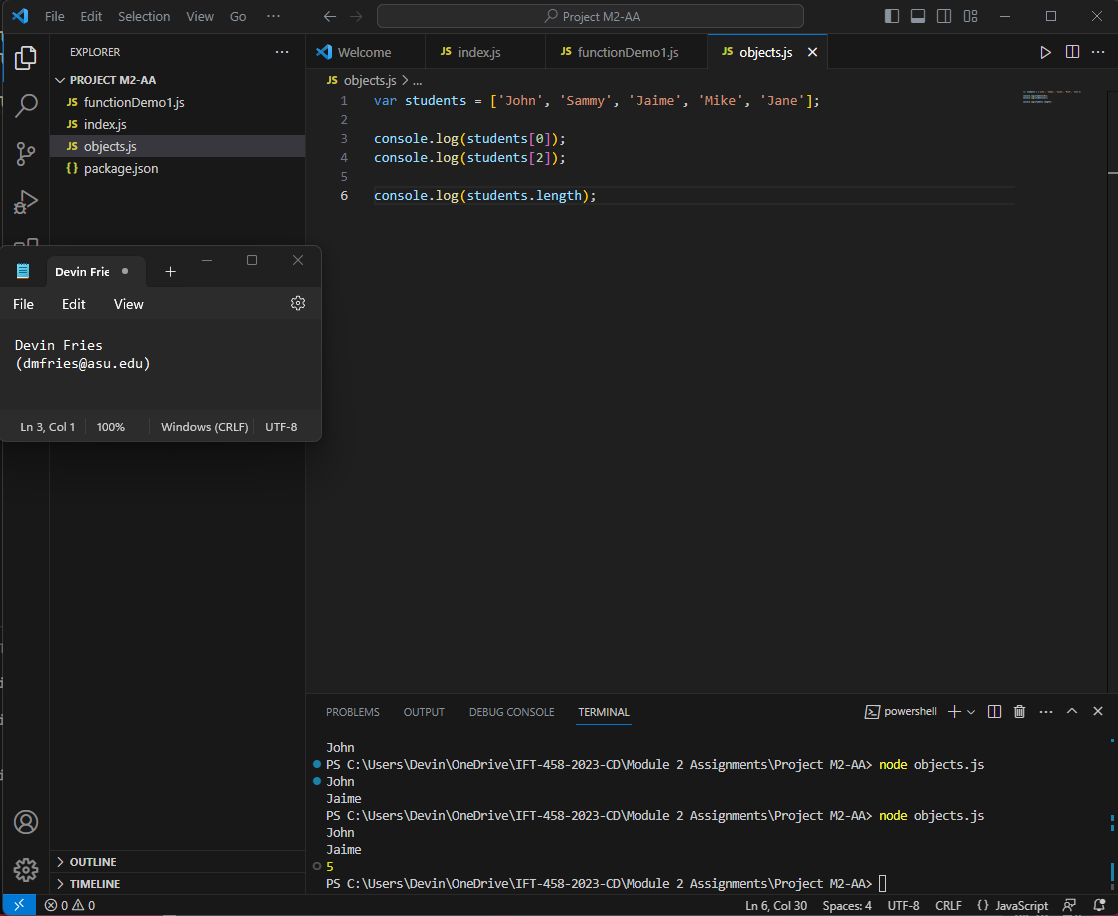
****

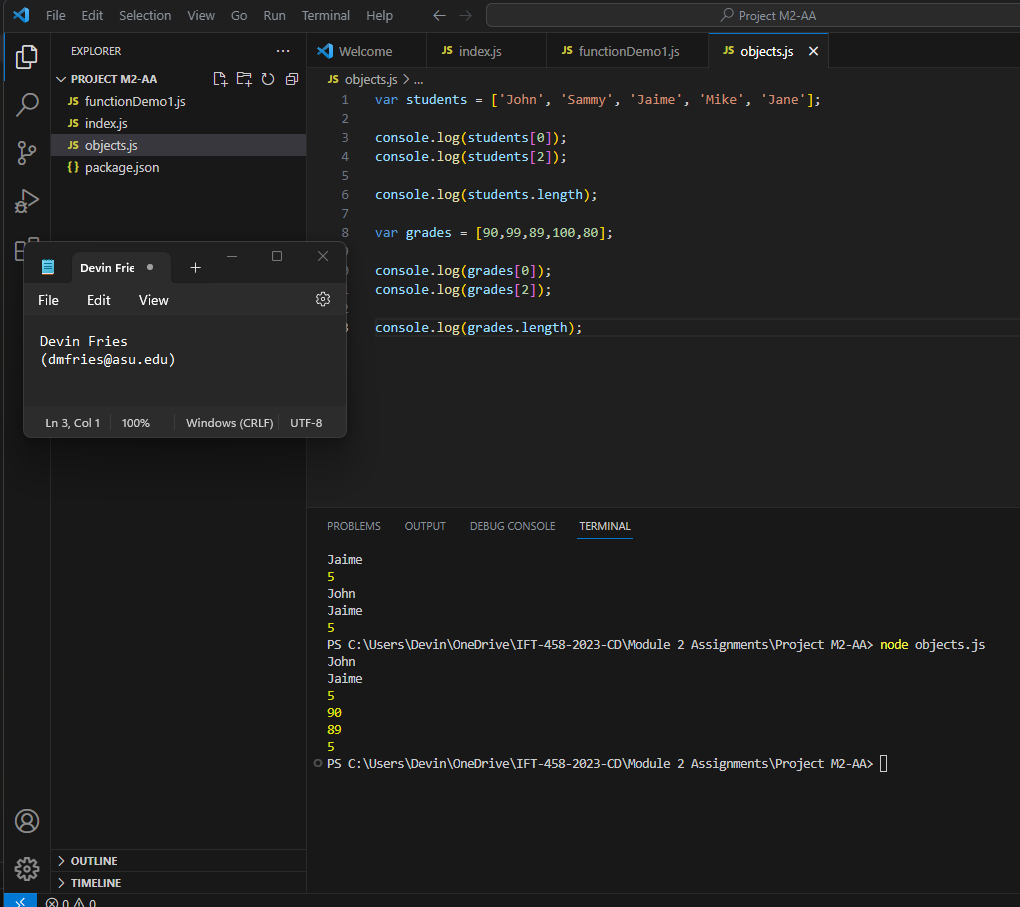




**Objects.js: **

****

****

****

**References**

EJS. (n.d.). Embedded JavaScript templating. EJS. Retrieved from <https://ejs.co/#features>