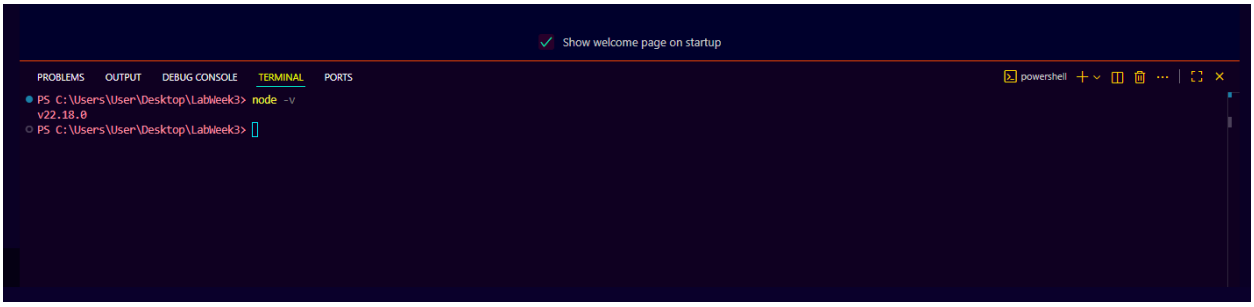
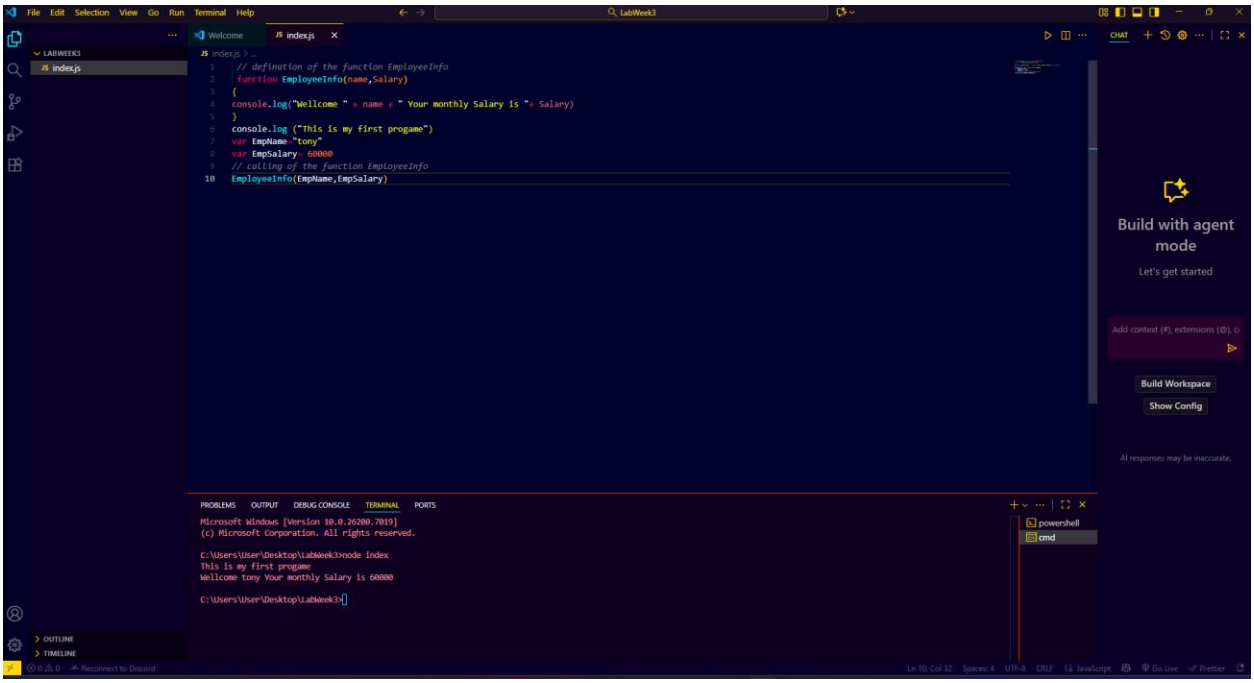
Cover Page

Exploring Functions, Arrow Functions, and NodeJS Modules

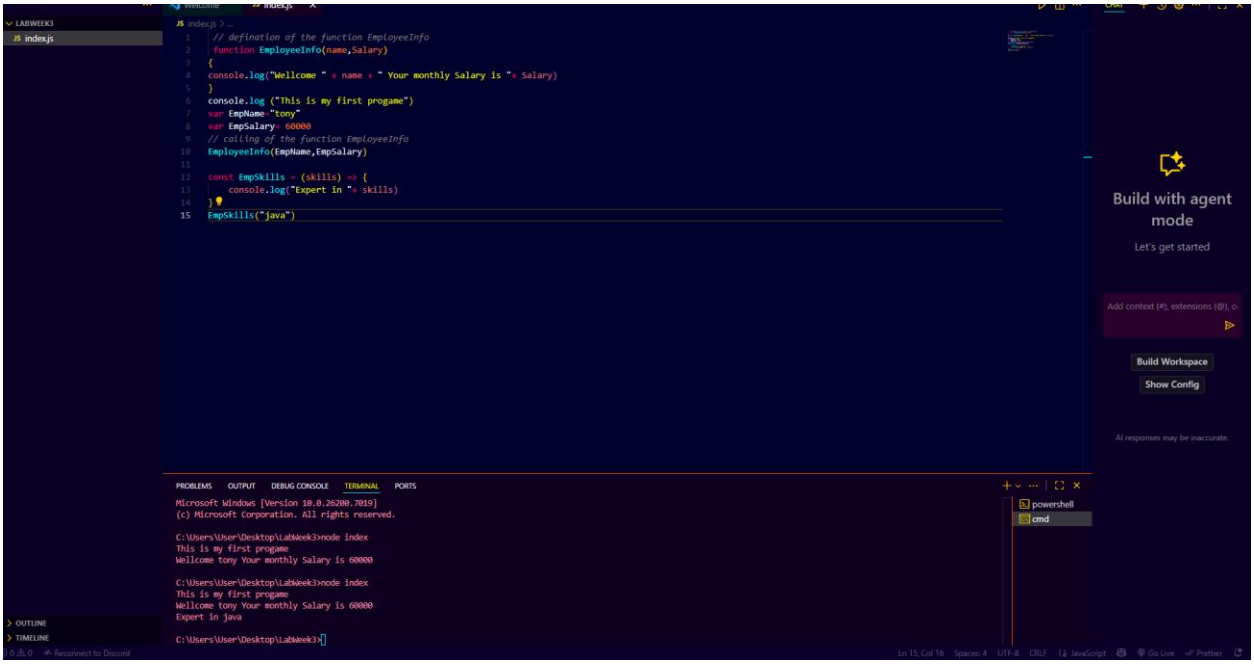
The lab was aimed at investigating the functionality of the modules in JavaScript with the help of Node.js. The goal was to come up with an idea of JavaScript functions, Arrow functions, local modules, and the core modules in Node. Having undergone four consecutive exercises, I studied how to create, export and import between certain files, and how to obtain system information using the in-built libraries of the Node. The experiment was conducted using Visual Studio Code, whereby v22.18.0 of Node has been installed and tested.



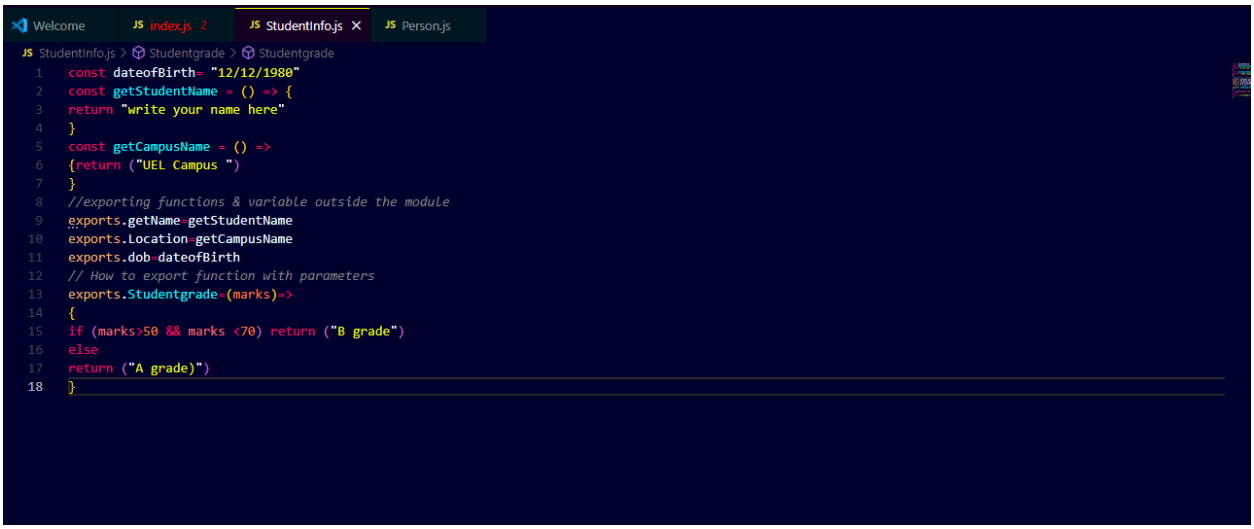
This was the first exercise that started with the creation of a new JavaScript file called index.js. I provided a simple routine named EmployeeInfo(name, Salary) which used the welcome message and the salary of the employee by using the statement of console.log. This drill revealed the fundamental form of a function in JavaScript, parameters use, and concatenation of the strings using the + operator. When the code was run on the command node index.js it was able to print the output This is my first program and the details of the employee.

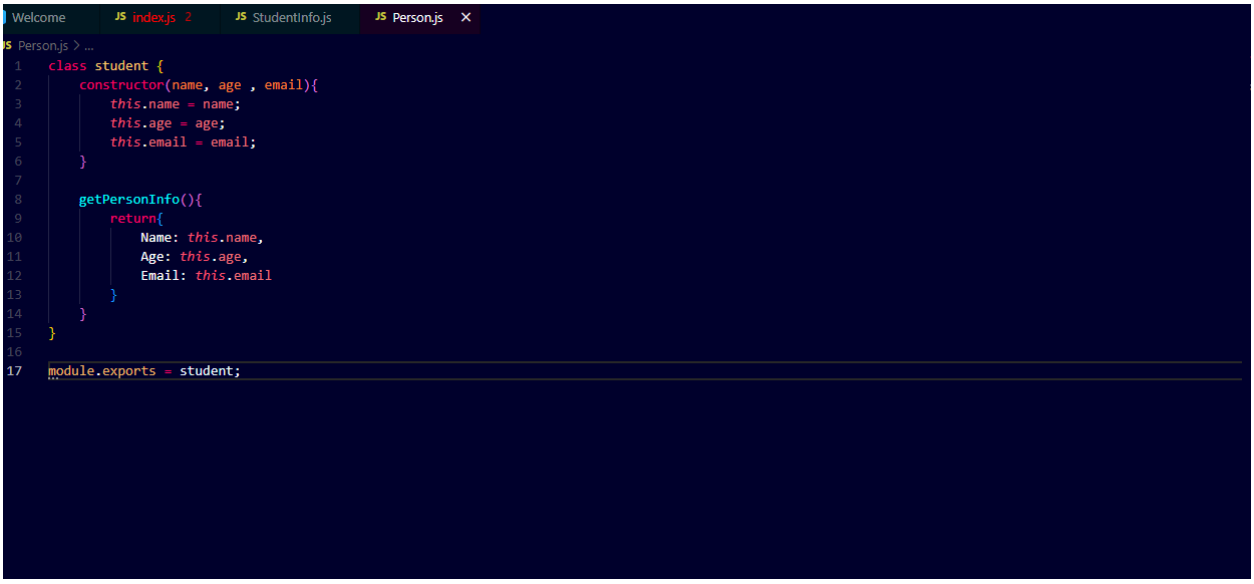


In Exercise 2, the assignment was to write an arrow function, a terse syntax of writing functions in ES6. I have included a new functionality named EmpSkills = (skills) => { console.log(knows Expert by the skills) in the same file. On execution, it printed Expert in Java, which meant that the arrow was defined and called the right way. The section of the lab allowed me to learn about the distinction between the traditional and arrow function syntax and how the arrow functions enhance the functionality and readability of modern JavaScript.

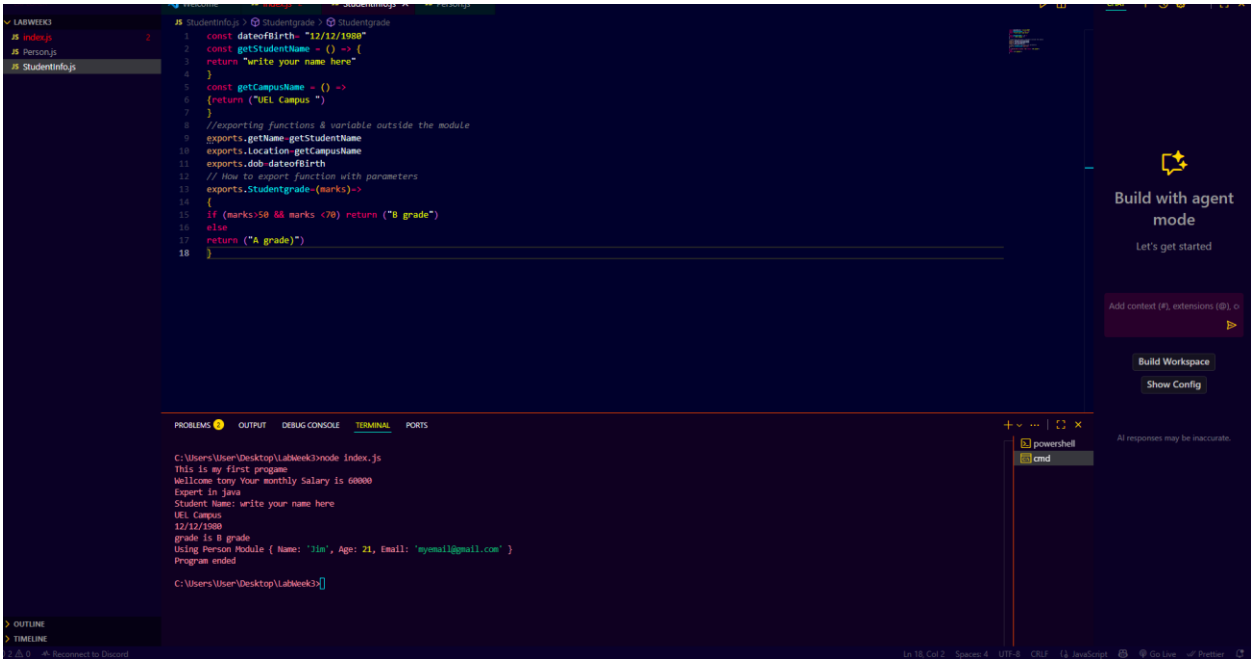


The third activity was the introduction of local modules in Node.js. I developed two different files called StudentInfo.js, Person.js (also known as EmployeeInfo and Person modules). I used variables and arrow functions, including getStudentName, getCampusName, and Studentgrade, in StudentInfo.js. All these were exported with the help of exports keyword so that they can be accessed in other files. Person.js In Person.js, I invoked the class and the constructor key words to create a class that contained name, age and email information about a person. The getPersonInfo was used to retrieve an object with all the personal information.



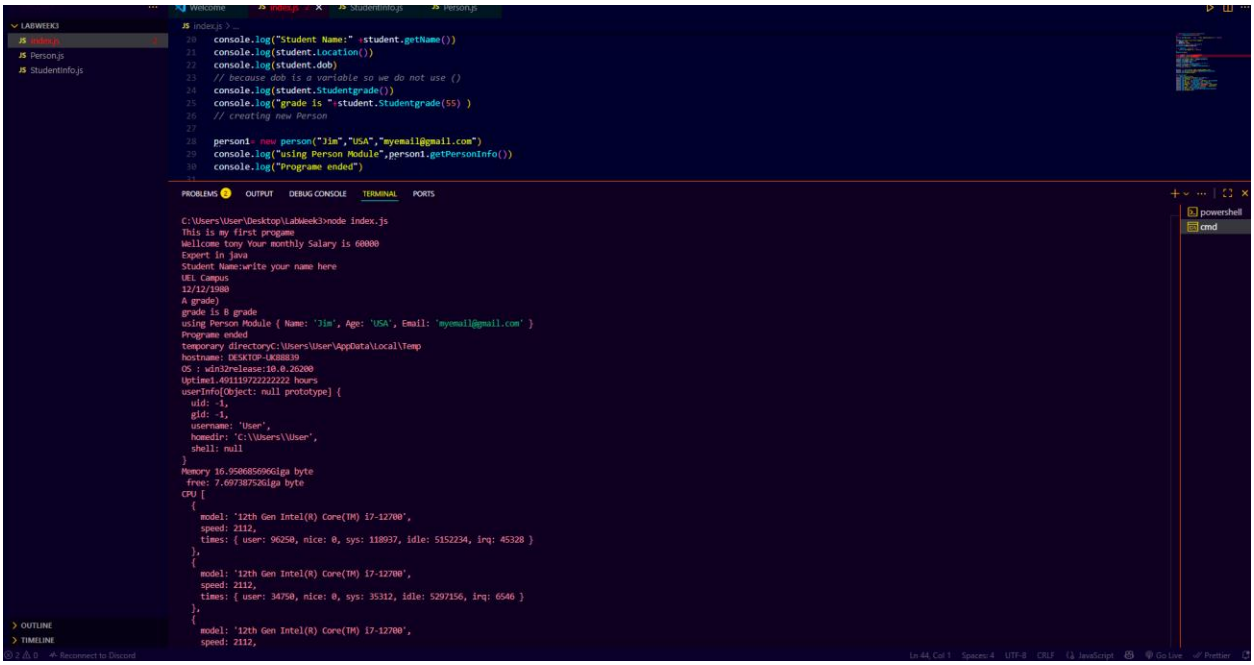


Once the two modules were created, I moved them to index.js using the require() function. The result of the output was that all the modules were functioning properly: it showed the name of the student, the campus where the student was situated, the date of birth, the grade, and it then constructed a Person object with the name, age, and email of the person. This practice allowed me to realize clearly the significance of modularity, reusability, and code organization in the development of a Node.js.



Lastly, Exercise 4 was dedicated to the use of the core modules of Node.js, including utility and OS. I used the following inbuilt libraries and also showed detailed system information such as temporary directory, hostname, operating system version, uptime, user information, total memory, free memory, CPU and network interfaces. This activity demonstrated the ability of the Node.js to communicate directly with the operating system of the computer and offer useful diagnostic or configuration information.

The experiment itself was successful and every code running did not have any syntax errors. Outputs in the terminal were the same as the ones shown by Dr. Nadeem Qazi in the lab guide. These exercises helped me obtain a realistic experience in defining functions, working with arrow syntax, modular programming, and utilizing the functionality of the Node.js core. I also felt freer using the terminal of Visual Studio Code to run JavaScript files with the help of the Node.



Finally, this lab has helped me gain a lot of knowledge about the use of modules to make JavaScript programs more structured and maintainable. Developers can use code to reuse and update elements through splitting code into several files. In addition, the use of the inbuilt modules of Node.js helped in demonstrating the versatility of JavaScript outside the browser context.