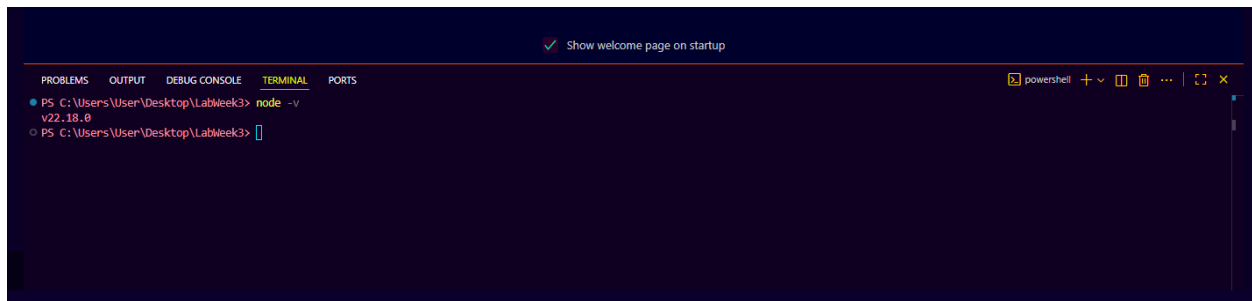


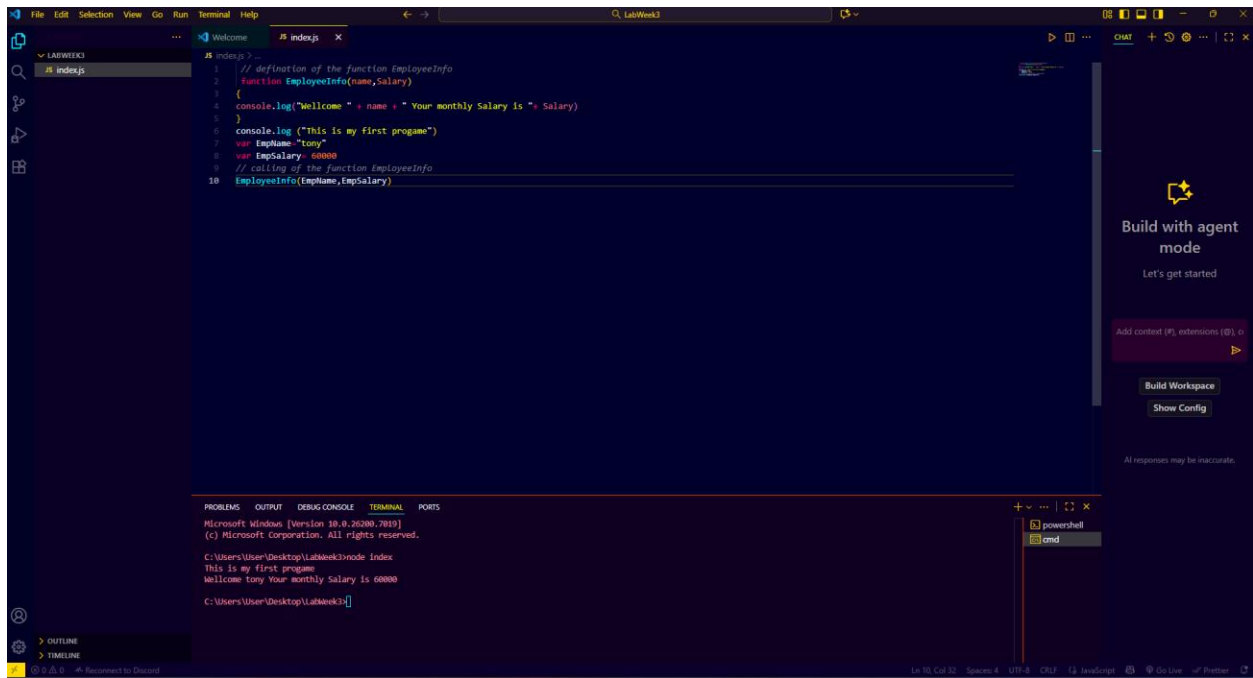
Cover Page

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.



The screenshot shows the Visual Studio Code editor interface. The main editor window displays a JavaScript file named `index.js` with the following code:

```
1 // definition of the function EmployeeInfo
2 function EmployeeInfo(name,Salary)
3 {
4   console.log("Welcome " + name + " Your monthly Salary is " + Salary)
5 }
6 console.log ("This is my first progame")
7 var EmpName = "tony"
8 var EmpSalary = 60000
9 // calling of the function EmployeeInfo
10 EmployeeInfo(EmpName,EmpSalary)
```

The bottom panel shows the `TERMINAL` tab with the following output:

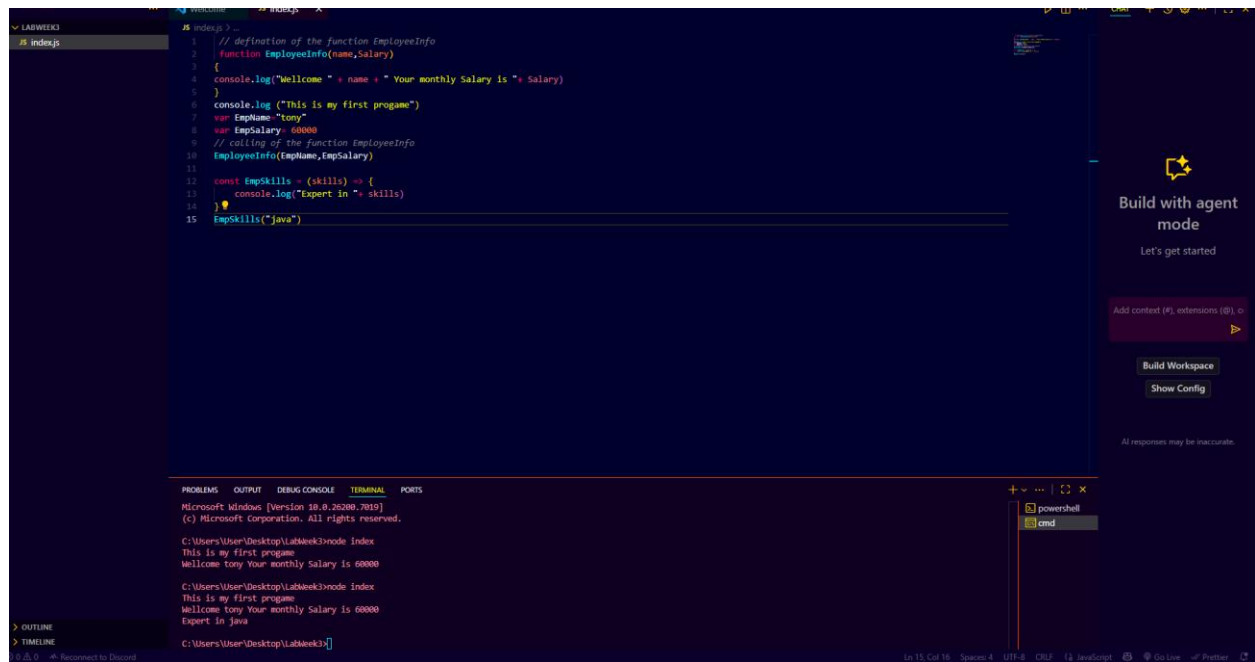
```
Microsoft Windows [Version 10.0.22000.7899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User\Desktop\LabWeek3>node index
This is my first progame
Welcome tony Your monthly Salary is 60000

C:\Users\User\Desktop\LabWeek3>
```

On the right side of the editor, there is a sidebar with a 'Build with agent mode' button and other options like 'Add context', 'Build Workspace', and 'Show Config'.

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.



```
1 // definition of the function EmployeeInfo
2 function EmployeeInfo(name,salary)
3 {
4   console.log("Welcome " + name + " Your monthly Salary is " + salary)
5 }
6 console.log ("This is my first progam")
7 var EmpName = "tony"
8 var EmpSalary= 60000
9 // calling of the function EmployeeInfo
10 EmployeeInfo(EmpName,EmpSalary)
11
12 const EmpSkills = (skills) => {
13   console.log("Expert in " + skills)
14 }
15 EmpSkills("java")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Microsoft Windows [Version 10.0.22000.7019]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User\Desktop\LabWeek3>node index
This is my first progam
Welcome tony Your monthly Salary is 60000

C:\Users\User\Desktop\LabWeek3>node index
This is my first progam
Welcome tony Your monthly Salary is 60000
Expert in java

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.

```
JS StudentInfo.js > JS Studentgrade > JS Studentgrade
1 const dateOfBirth = "12/12/1980"
2 const getStudentName = () => {
3   return "write your name here"
4 }
5 const getCampusName = () => {
6   return ("UEL Campus")
7 }
8 //exporting functions & variable outside the module
9 exports.getName = getStudentName
10 exports.Location = getCampusName
11 exports.dob = dateOfBirth
12 // How to export function with parameters
13 exports.Studentgrade = (marks) => {
14   {
15     if (marks > 50 && marks < 70) return ("B grade")
16     else
17       return ("A grade")
18   }
19 }
```

```
JS Person.js > ...
1 class student {
2   constructor(name, age, email){
3     this.name = name;
4     this.age = age;
5     this.email = email;
6   }
7
8   getPersonInfo(){
9     return{
10       Name: this.name,
11       Age: this.age,
12       Email: this.email
13     }
14   }
15 }
16
17 module.exports = student;
```

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.

The screenshot shows the Visual Studio Code interface. The editor window displays a JavaScript file named `StudentInfo.js` with the following code:

```
1  <const dateofBirth= "12/12/1980">
2  <const getStudentName = () => {
3  <return "write your name here">
4  <}>
5  <const getCampusName = () => {
6  <return ("UEL Campus")>
7  <}>
8  //exporting functions & variable outside the module
9  <exports.getName=getStudentName>
10 <exports.getLocation=getCampusName>
11 <exports.dob=dateofBirth>
12 // How to export function with parameters
13 <exports.Studentgrade=(marks)->{
14 <{
15 <  if (marks>50 && marks <70) return ("B grade")
16 <  else
17 <    return ("A grade")
18 <}>
19 <}>
```

The terminal window at the bottom shows the output of running the code:

```
C:\Users\User\Desktop\LabWeek3>node index.js
This is my first program
Welcome tony Your monthly Salary is 60000
Expert in java
Student Name: write your name here
UEL Campus
12/12/1980
grade is B grade
Using Person Module { Name: 'Jin', Age: 21, Email: 'myemail@gmail.com' }
Program ended
C:\Users\User\Desktop\LabWeek3>
```

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.

```
20 console.log("Student Name:" + student.getName())
21 console.log(student.location())
22 console.log(student.dob)
23 // because dob is a variable so we do not use ()
24 console.log(student.Studentgrade())
25 console.log("grade is " + student.Studentgrade(ss) )
26 // creating new Person
27
28 person1= new person("Jia","USA","myemail@gmail.com")
29 console.log("using Person Module",person1.getPersonInfo())
30 console.log("Program ended")
31
```

```
C:\Users\User\Desktop\LabWeek3>node index.js
This is my first program
Welcome to my Your monthly Salary is 60000
Expert in Java
Student Name:write your name here
UCL Campus
12/12/1980
A grade)
grade is B grade
using Person Module { Name: 'Jia', Age: 'USA', Email: 'myemail@gmail.com' }
Program ended
Temporary directory:C:\Users\User\AppData\Local\Temp
hostname: DESKTOP-UMH80
OS : win32release:10.0.20200
Uptime:1.4911972222222222 hours
userInfo(Object: null prototype) {
  uid: -1,
  gid: -1,
  username: 'User',
  homedir: 'c:\Users\User',
  shell: null
}
Memory 10.920805290610 byte
free: 7.69730730610 byte
CPU [
  {
    model: '12th Gen Intel(R) Core(TM) i7-12700',
    speed: 2112,
    times: { user: 96250, nice: 0, sys: 118937, idle: 5152234, irq: 45328 }
  },
  {
    model: '12th Gen Intel(R) Core(TM) i7-12700',
    speed: 2112,
    times: { user: 34750, nice: 0, sys: 35312, idle: 5297156, irq: 6540 }
  },
  {
    model: '12th Gen Intel(R) Core(TM) i7-12700',
    speed: 2112,

```

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.