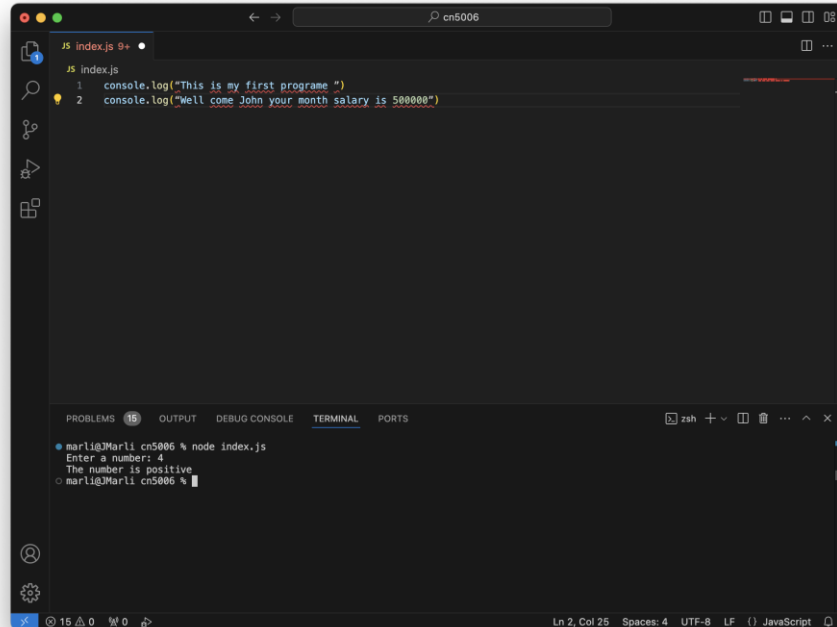


U2283556

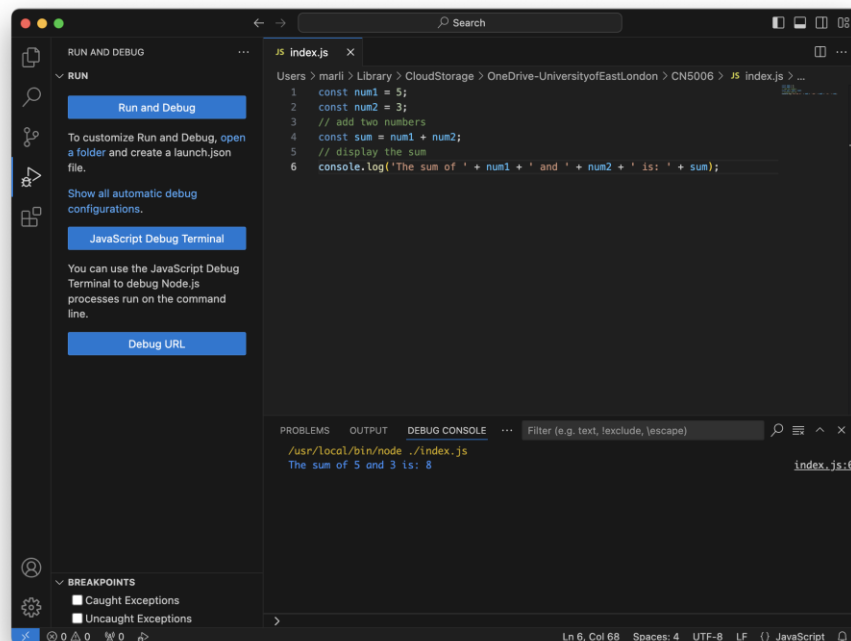
# CN5006 PORTFOLIO:

## Week 1 code:



```
JS index.js 9+
1 console.log("This is my first programe ")
2 console.log("Well come John your month salary is 500000")

PROBLEMS 15 OUTPUT DEBUG CONSOLE TERMINAL PORTS
marli@Marli cn5006 % node index.js
Enter a number: 4
The number is positive
marli@Marli cn5006 %
```



```
JS index.js x
1 const num1 = 5;
2 const num2 = 3;
3 // add two numbers
4 const sum = num1 + num2;
5 // display the sum
6 console.log("The sum of " + num1 + " and " + num2 + " is: " + sum);

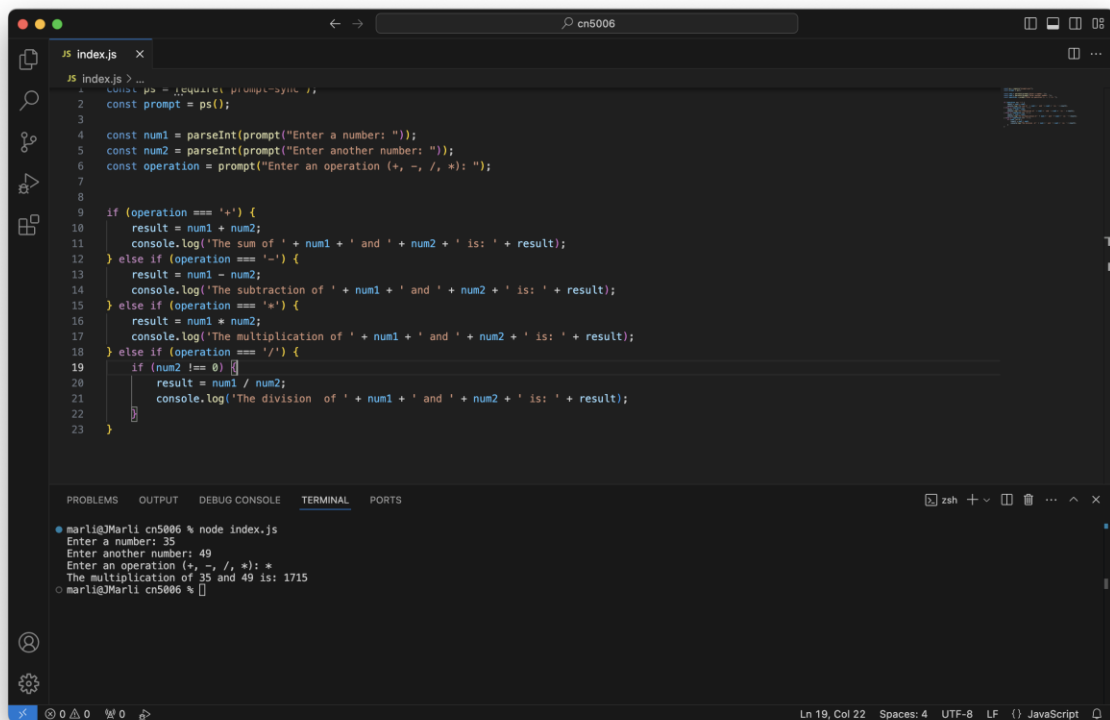
PROBLEMS OUTPUT DEBUG CONSOLE ... Filter (e.g. text, exclude, escape)
/usr/local/bin/node ./index.js
The sum of 5 and 3 is: 8 index.js:6
```

## Self-evaluation:

I have improved my java script language skills and will work more to improve them. This was a great refresher after the summer holiday.

U2283556

## Exercise:



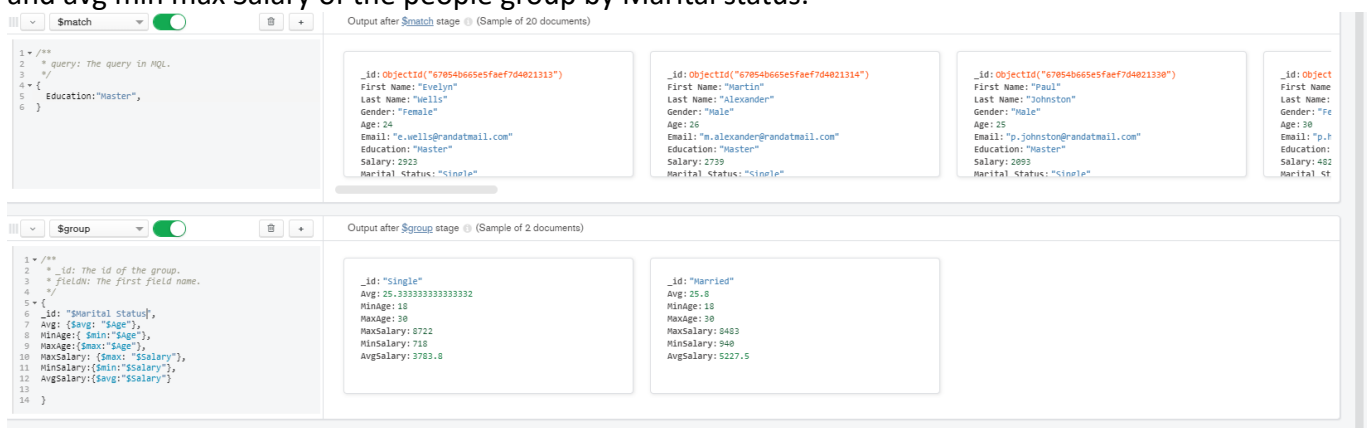
```
1 // index.js
2 const ps = (prompt) => {
3   const prompt = ps();
4   const num1 = parseInt(prompt("Enter a number: "));
5   const num2 = parseInt(prompt("Enter another number: "));
6   const operation = prompt("Enter an operation (+, -, /, *): ");
7
8   if (operation === '+') {
9     result = num1 + num2;
10    console.log('The sum of ' + num1 + ' and ' + num2 + ' is: ' + result);
11  } else if (operation === '-') {
12    result = num1 - num2;
13    console.log('The subtraction of ' + num1 + ' and ' + num2 + ' is: ' + result);
14  } else if (operation === '*') {
15    result = num1 * num2;
16    console.log('The multiplication of ' + num1 + ' and ' + num2 + ' is: ' + result);
17  } else if (operation === '/') {
18    if (num2 !== 0) {
19      result = num1 / num2;
20      console.log('The division of ' + num1 + ' and ' + num2 + ' is: ' + result);
21    }
22  }
23 }
```

```
marli@Marli cn5006 % node index.js
Enter a number: 35
Enter another number: 49
Enter an operation (+, -, /, *): *
The multiplication of 35 and 49 is: 1715
marli@Marli cn5006 %
```

This program was created to function as a calculator that is capable of four kinds of operations.

## Week 2 lab tasks:

1) Repeat the same process to search Education for Master and .Find the avg,min,max age and avg min max Salary of the people group by Marital status.



**\$match** stage (Sample of 20 documents)

```
1 // **
2 * query: The query in MQL.
3 *
4 * {
5   Education: "Master",
6 }
```

Document
<pre>{   "_id": "67054b665e5faef7d4021313",   "First Name": "Evelyn",   "Last Name": "Wells",   "Gender": "Female",   "Age": 24,   "Email": "e.wells@randatmail.com",   "Education": "Master",   "Salary": 2923,   "Marital Status": "Single" }</pre>
<pre>{   "_id": "67054b665e5faef7d4021314",   "First Name": "Martin",   "Last Name": "Alexander",   "Gender": "Male",   "Age": 26,   "Email": "m.alexander@randatmail.com",   "Education": "Master",   "Salary": 2723,   "Marital Status": "Single" }</pre>
<pre>{   "_id": "67054b665e5faef7d4021330",   "First Name": "Paul",   "Last Name": "Johnston",   "Gender": "Male",   "Age": 25,   "Email": "p.johnston@randatmail.com",   "Education": "Master",   "Salary": 2093,   "Marital Status": "Single" }</pre>
<pre>{   "_id": "67054b665e5faef7d4021331",   "First Name": "John",   "Last Name": "Johnston",   "Gender": "Male",   "Age": 30,   "Email": "j.johnston@randatmail.com",   "Education": "Master",   "Salary": 442,   "Marital Status": "Single" }</pre>

**\$group** stage (Sample of 2 documents)

```
1 // **
2 * _id: The id of the group.
3 * fields: The first field name.
4 *
5 * {
6   _id: "$marital status",
7   avg: {$avg: "$age"},
8   minAge: {$min: "$age"},
9   maxAge: {$max: "$age"},
10  maxSalary: {$max: "$salary"},
11  minSalary: {$min: "$salary"},
12  avgSalary: {$avg: "$salary"}
13 }
14
```

Document
<pre>{   "_id": "Single",   "avg": 25.333333333333332,   "minAge": 18,   "maxAge": 30,   "minSalary": 8722,   "maxSalary": 718,   "avgSalary": 3783.8 }</pre>
<pre>{   "_id": "Married",   "avg": 25.0,   "minAge": 18,   "maxAge": 30,   "minSalary": 8403,   "maxSalary": 940,   "avgSalary": 5227.5 }</pre>

U2283556

## 2) 2. find min,max average salary of each age group of female

The screenshot shows the MongoDB Compass interface. The top stage is a **\$match** stage with a query: `{ "gender": "Female" }`. The output shows 20 documents. The bottom stage is a **\$group** stage with a query: `{ "_id": "$age", "avgSalary": { "$avg": "$salary" }, "minSalary": { "$min": "$salary" }, "maxSalary": { "$max": "$salary" } }`. The output shows 13 documents grouped by age (19, 24, 26, 28).

Age Group	minSalary	avgSalary	maxSalary
19	516	5090.909090909091	5945
24	3529	4329	6921
26	4490	4412.4	8170
28	1150	3333.333333333333	3529

## 3) find min,max average salary of each age group of male

The screenshot shows the MongoDB Compass interface. The top stage is a **\$match** stage with a query: `{ "gender": "Male" }`. The output shows 20 documents. The bottom stage is a **\$group** stage with a query: `{ "_id": "$age", "avgSalary": { "$avg": "$salary" }, "minSalary": { "$min": "$salary" }, "maxSalary": { "$max": "$salary" } }`. The output shows 13 documents grouped by age (19, 24, 26, 28).

Age Group	minSalary	avgSalary	maxSalary
19	516	5090.909090909091	5945
24	3529	4329	6921
26	4490	4412.4	8170
28	1150	3333.333333333333	3529

## 4) Count married and unmarried females and males.

The screenshot shows the MongoDB Compass interface. The top stage is a **\$match** stage with a query: `{ "gender": "Female", "maritalStatus": "Married" }`. The output shows 20 documents. The bottom stage is a **\$group** stage with a query: `{ "_id": "$gender", "$sum": { "$sum": "$count" } }`. The output shows 2 documents: `{ "_id": "female", "$sum": 42 }` and `{ "_id": "male", "$sum": 58 }`.

Gender	Count
female	42
male	58

U2283556

The screenshot shows the MongoDB Compass web interface. At the top, there are tabs for Documents, Aggregations (selected), Schema, Explain Plan, Indexes, and Validation. The top right corner displays statistics: DOCUMENTS 200, TOTAL SIZE 38.1KB, AVG. SIZE 190B, INDEXES 1, TOTAL SIZE 36.0KB, and AVG. SIZE 36.0KB. The main workspace is divided into three sections. The top section shows a collection of documents with fields like First Name, Last Name, Gender, Age, Email, Education, Salary, and Marital Status. The middle section shows the output after the \$match stage, which filters documents where Marital Status is 'single'. The bottom section shows the output after the \$group stage, which groups documents by gender and calculates the count for each group. The aggregation pipeline is defined in the left sidebar:

```
1 // **
2 * query: The query in JSON.
3 *
4 *
5 {
6   "Marital Status": "single",
7 }
8
9
10
```

The \$match stage is configured with the following query:

```
{
  "Marital Status": "single"
}
```

The \$group stage is configured with the following query:

```
{
  "_id": "$gender",
  "count": {
    "$sum": 1
  }
}
```

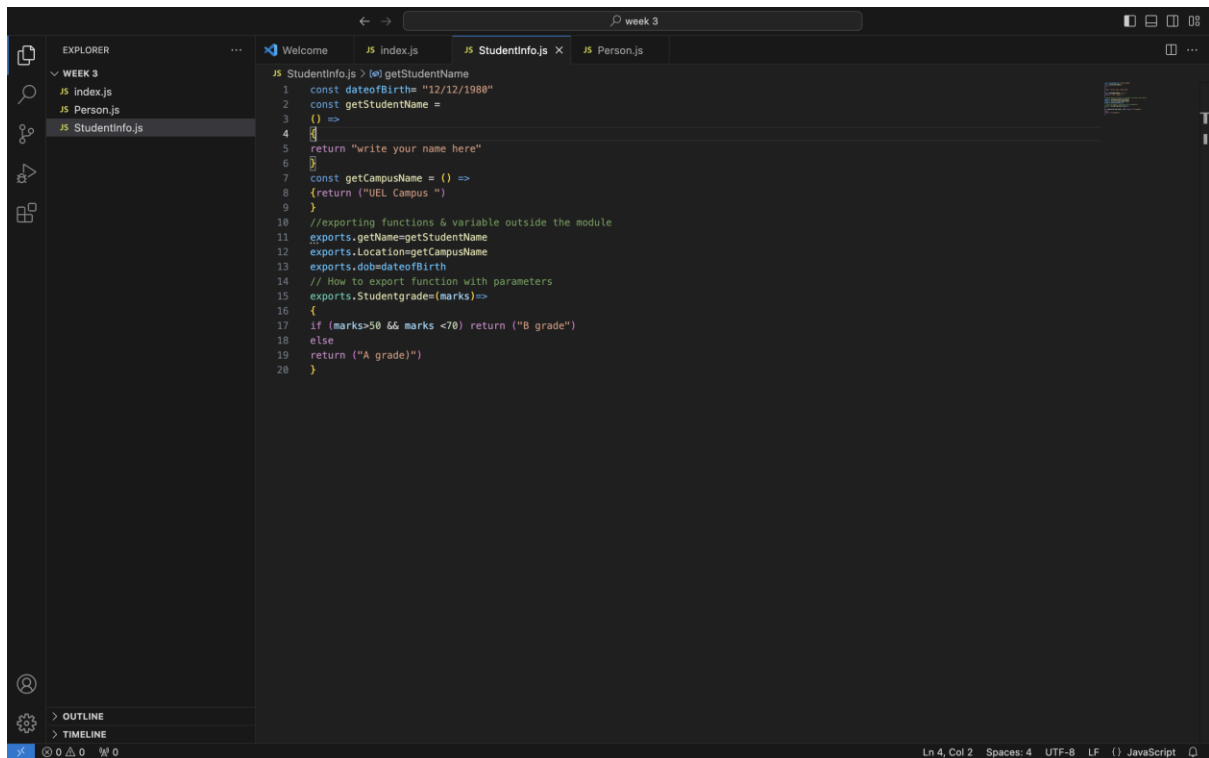
## Report:

The work that has been done with this code has been used to separate the data that has been given from a spreadsheet then sorted into different categories depending on the need.

## WEEK 3:

Submit the code for the completed Exercises i.e.: i) Index.js ii) Person.js iii) Employeeinfo.js iv) Exercise 4.js

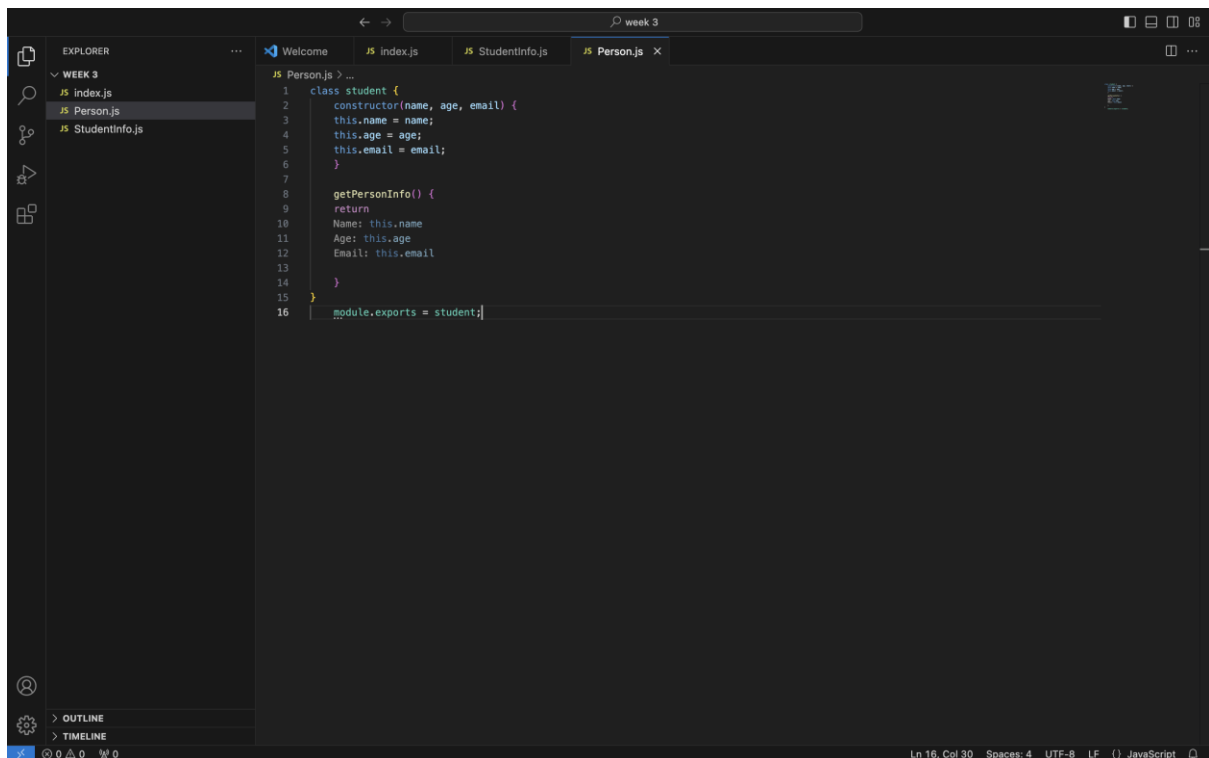
U2283556



This screenshot shows the Visual Studio Code editor with the file `StudentInfo.js` open. The Explorer sidebar on the left shows a project structure for 'WEEK 3' containing `index.js`, `Person.js`, and `StudentInfo.js`. The main editor area displays the following JavaScript code:

```
1  const dateOfBirth = "12/12/1980"
2  const getStudentName = () =>
3  {
4    // Write your name here
5    return "write your name here"
6  }
7  const getCampusName = () =>
8  {
9    return ("UEL Campus ")
10 }
11 //exporting functions & variable outside the module
12 exports.getName=getStudentName
13 exports.Location=getCampusName
14 exports.dob=dateOfBirth
15 // How to export function with parameters
16 exports.Studentgrade=(marks)=>
17 {
18   if (marks>50 && marks <70) return ("B grade")
19   else
20     return ("A grade")
21 }
```

The status bar at the bottom indicates the cursor is at Line 4, Column 2, with 4 spaces, in UTF-8 encoding, LF line endings, and JavaScript syntax.

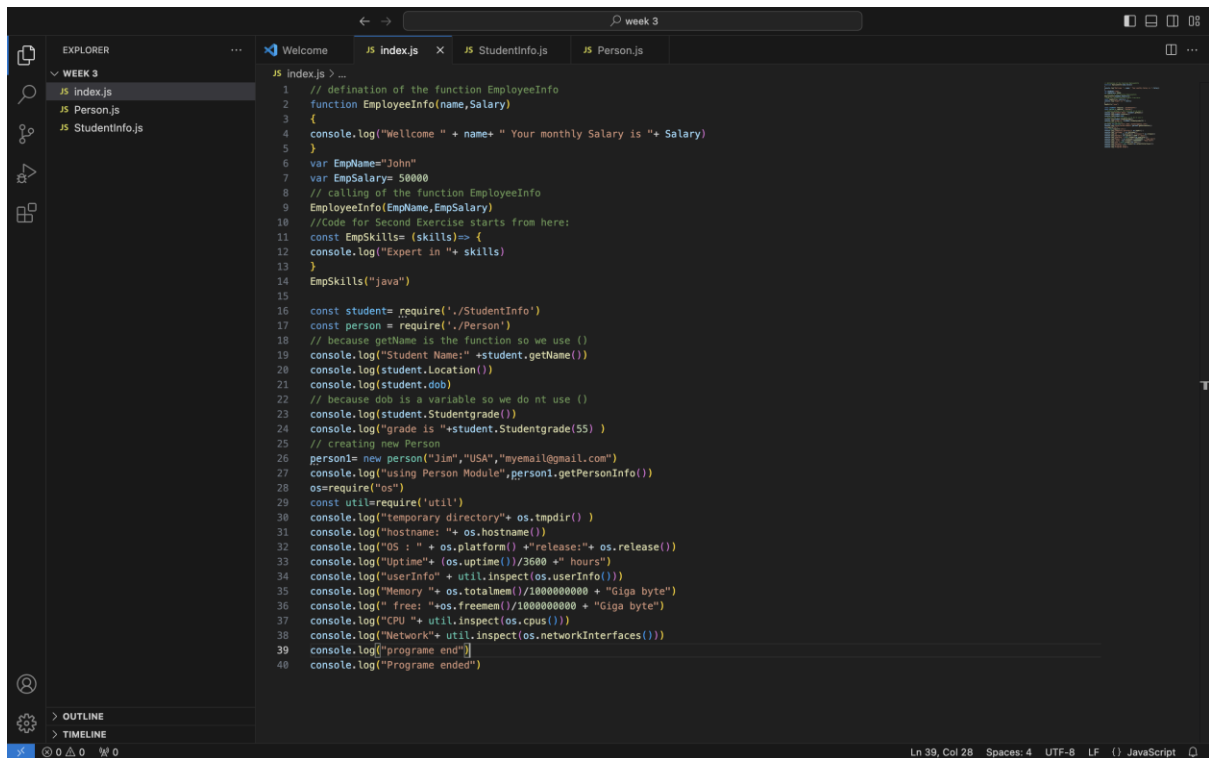


This screenshot shows the Visual Studio Code editor with the file `Person.js` open. The Explorer sidebar on the left shows the same project structure as the previous screenshot. The main editor area displays the following JavaScript code:

```
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     {
11       Name: this.name,
12       Age: this.age,
13       Email: this.email
14     }
15   }
16   module.exports = student;
```

The status bar at the bottom indicates the cursor is at Line 16, Column 30, with 4 spaces, in UTF-8 encoding, LF line endings, and JavaScript syntax.

U2283556



```
1 // definition of the function EmployeeInfo
2 function EmployeeInfo(name,Salary)
3 {
4   console.log("Wellcome " + name+ " Your monthly Salary is "+ Salary)
5 }
6 var EmpName="John"
7 var EmpSalary= 50000
8 // calling of the function EmployeeInfo
9 EmployeeInfo(EmpName,EmpSalary)
10 //Code for Second Exercise starts from here:
11 const EmpSkills= (skills)=> {
12   console.log("Expert in "+ skills)
13 }
14 EmpSkills("java")
15
16 const student= require('./StudentInfo')
17 const person = require('./Person')
18 // because getName is the function so we use ()
19 console.log("Student Name:" +student.getName())
20 console.log(student.Location())
21 console.log(student.dob)
22 // because dob is a variable so we do nt use ()
23 console.log(student.Studentgrade())
24 console.log("grade is "+student.Studentgrade(55) )
25
26 // creating new Person
27 person= new person("Jim","USA","myemail@gmail.com")
28 console.log("using Person Module",person1.getPersonInfo())
29 os=require("os")
30 const util=require('util')
31 console.log("temporary directory"+ os.tmpdir() )
32 console.log("hostname: " + os.hostname())
33 console.log("OS : " + os.platform() + "release:"+ os.release())
34 console.log("Uptime"+ (os.uptime())/3600 + " hours")
35 console.log("userInfo" + util.inspect(os.userInfo()))
36 console.log("Memory "+ os.totalmem()/1000000000 + "Giga byte")
37 console.log(" free: "+os.freemem()/1000000000 + "Giga byte")
38 console.log("CPU "+ util.inspect(os.cpus()))
39 console.log("Network"+ util.inspect(os.networkInterfaces()))
40 console.log("Programme end")
41 console.log("Programme ended")
```

## Report:

The coding that I've done during this lesson displays the information of the device that is being used. It also displays the information of the student information that has been given. I learned how to link different methods together.