Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №1

По дисциплине: Архитектура ЭВМ

Тема: Знакомство с Node.JS

Студент Чаушев А.К..

Группа ИУ7-56Б

Оценка (баллы) _____

Преподаватели Попов А. Ю.

Москва — 2020 г.

1 Введение

1.1 Цель работы

Цель лабораторной работы являеется ознакомление с Node.js, Работа с целыми числами, Циклы, Массивы, Строки, Объекты, Ссылочний типы данных, Функции.

2 Задания (TASK 1)

Задание 1 Создать хранилище в оперативной памяти для хранения информации о детях. Необходимо хранить информацию о ребенке: фамилия и возраст.Необходимо обеспечить уникальность фамилий детей.
Реализовать функции:

1. CREATE READ UPDATE DELETE для детей в хранилище
2. Получение среднего возраста детей
3. Получение информации о самом старшем ребенке
4. Получение информации о детях, возраст которых входит в заданный отрезок
5. Получение информации о детях, фамилия которых начинается с заданной буквы
6. Получение информации о детях, фамилия которых длиннее заданного количества символов
7. Получение информации о детях, фамилия которых начинается с гласной буквы

Задание 2 Создать хранилище в оперативной памяти для хранения информации о студентах. Необходимо хранить информацию о студенте: название группы, номер студенческого билета, оценки по программированию.Необходимо обеспечить уникальность номеров студенческих билетов.

Реализовать функции:

1. CREATE READ UPDATE DELETE для студентов в хранилище
2. Получение средней оценки заданного студента
3. Получение информации о студентах в заданной группе
4. Получение студента, у которого наибольшее количество оценок в заданной группе
5. Получение студента, у которого нет оценок

Задание 3 Создать хранилище в оперативной памяти для хранения точек. Неоходимо хранить информацию о точке: имя точки, позиция X и позиция Y. Необходимо обеспечить уникальность имен точек.

Реализовать функции:

1. CREATE READ UPDATE DELETE для точек в хранилище
2. Получение двух точек, между которыми наибольшее расстояние
3. Получение точек, находящихся от заданной точки на расстоянии, не превышающем заданную константу
4. Получение точек, находящихся выше / ниже / правее / левее заданной оси координат
5. Получение точек, входящих внутрь заданной прямоугольной зоны

3 Листинг (Task 1)

3.1 Задание 1

```
"use strict"

function isVowel(char)
{
  if (char.length == 1)
  {
    var vowels = new Array('a','e','i','o','u', 'A', 'E', 'I', 'O', 'U');
    var isVowel = false;

    for(let e in vowels)
    {
      if(vowels[e] == char)
      {
        isVowel = true;
      }
    }

    return isVowel;
  }
}

class ChildrenRepository {

    constructor() {
        this.dict = {};
    }

    addChild(surname, age) {
        if (this.dict[surname]) {
            return;
        }
        this.dict[surname] = age;
    }

    deleteChild(surname){
        if (this.dict[surname]) {
            delete this.dict[surname];
        }
    }

    getChild(surname) {
        if (this.dict[surname]) {
            return this.dict[surname];
        }
    }

    getAllChildren() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                console.log("Child: " + key + " " + "Age: " + this.dict[key]);
            }
```

```
        }
    }

    updateAge(surname , age) {
        if (this.dict[surname]) {
            this.dict[surname] = age
        }
    }

    findAvarageAge() {
        let result = 0;
        let count = 0;
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                count++;
                result += this.dict[key];
            }
        }
        if (count != 0) {
            return result/count;
        } else{
            return 0;
        }
    }

    findOldest() {
        let oldestKey = null;
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (oldestKey == null) {
                    oldestKey = key;
                } else if (this.dict[key] > this.dict[oldestKey]) {
                    oldestKey = key;
                }
            }
        }

        console.log("Child: " + oldestKey + " " + "Age: " + this.dict[oldestKey]);
    }

    findInRange(from, to) {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key] > from && this.dict[key] < to) {
                    console.log("Child: " + key + " " + "Age: " + this.dict[key]);
                }
            }
        }
    }

    findFirstSymbolName(symbol) {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
               if (key[0] == symbol) {
                console.log("Child: " + key + " " + "Age: " + this.dict[key]);
               }
            }
        }
    }

    findNameLongerThan(value) {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
               if (key.length > value) {
                console.log("Child: " + key + " " + "Age: " + this.dict[key]);
               }
            }
        }
    }

    findNameStartOnVowel() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
               if (isVowel(key[0])){
                console.log("Child: " + key + " " + "Age: " + this.dict[key]);
               }
            }
        }
    }
}
```

3.2 Задание 2

```
    "use strict"

class Student {

    constructor(group, number, grades) {
```

```
        this.group = group;
        this.number = number;
        this.grades = grades;
    }

    changeGroup(group) {
        this.group = group;
    }

    findAvarageGrade() {
        let count = 0;
        let result = 0;
        for(let i=0; i < this.grades.length; i++) {
            result += this.grades[i];
            count++;
        }

        if (count != 0) {
            return result/count;
        } else{
            return 0;
        }
    }
}

class StudentsDB {

    constructor() {
        this.dict = {};
    }

    addStudent(number, student) {
        if (this.dict[number]) {
            return;
        }
        this.dict[number] = student;
    }

    findAllStudents() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                console.log(this.dict[key]);
            }
        }
    }

    updateStudent(number, student) {
        this.dict[number] = student;
    }

    deleteStudent(number){
        if (this.dict[number]) {
            delete this.dict[number];
        }
    }

    getAvarageGradeOfStudent(number) {
        return this.dict[number].findAvarageGrade();
    }

    findStudentsByGroup(group) {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key].group == group)
                    console.log(this.dict[key]);
            }
        }
    }

    findStudentWithMostGradesByGroup(group) {
        let mostGrades = null;
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key].group == group)
                    if (mostGrades == null) {
                        mostGrades = this.dict[key].number;
                    } else {
                        if (this.dict[mostGrades].grades.length < this.dict[key].grades.length) {
                            mostGrades = key;
                        }
                    }
            }
        }

        console.log(this.dict[mostGrades]);
    }

    findStudentsWithoutGrades() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key].grades.length == 0) {
```

```
                    console.log(this.dict[key])
                }
            }
        }

    }
}
```

3.3 Задние 3

```
    "use strict"

class Point {
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }
}

class PointRepository {
    constructor() {
        this.dict = {}
    }

    addPoint(name, point) {
        if (this.dict[name]) {
            return;
        }
        this.dict[name] = point;
    }

    findPoint(name) {
        console.log(name + ":");
        console.log(this.dict[name]);
    }

    findAllPoints(){
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                console.log(key + ":");
                console.log(this.dict[key]);
            }
        }
    }

    updatePoint(name) {
        this.dict[name] = point;
    }

    deletePoint(name) {
        if (this.dict[name]) {
            delete this.dict[name];
        }
    }

    findDistanceBetweenPoints(point1, point2) {
        let distance = Math.sqrt((point1.x - point2.x) * (point1.x - point2.x) +
        (point1.y - point2.y) * (point1.y - point2.y));

        return distance;
    }

    findTwoFarestPoints() {
        let distance = null;
        let point1 = null;
        let point2 = null;
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                for (let key2 in this.dict) {
                    if (this.dict.hasOwnProperty(key2)) {
                        if (distance = null) {
                            distance = this.findDistanceBetweenPoints(this.dict[key], this.dict[key2]);
                            point1 = this.dict[key];
                            point2 = this.dict[key2];
                        }else{
                            let temp = this.findDistanceBetweenPoints(this.dict[key], this.dict[key2]);
                            console.log(temp);
                            if (temp > distance) {
                                distance = temp;
                                point1 = this.dict[key];
                                point2 = this.dict[key2];
                            }
                        }
                    }
                }
            }
        }
        console.log(point1);
```

```
        console.log(point2);
    }

    findPointFromAtRange(point, range) {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                let distance = this.findDistanceBetweenPoints(this.dict[point], this.dict[key]);
                //console.log(distance)
                if (distance <= range) {
                    console.log(this.dict[key]);
                }
            }
        }
    }

    findPointsAboveOx() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key].y > 0) {
                    console.log(this.dict[key]);
                }
            }
        }
    }

    findPointsUnderOx() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key].y < 0) {
                    console.log(this.dict[key]);
                }
            }
        }
    }

    findPointsLeftFromY() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key].x < 0) {
                    console.log(this.dict[key]);
                }
            }
        }
    }

    findPointsRightFromY() {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.dict[key].x > 0) {
                    console.log(this.dict[key]);
                }
            }
        }
    }

    // Note it should be rectangle 2 points bottom-left and top-right
    isInRectangle(x1, y1, x2, y2, x, y) {
        if (x > x1 && x < x2 &&
            y > y1 && y < y2)
            return true;

        return false;
    }

    findPointsInRectangle(x1, y1, x2, y2) {
        for (let key in this.dict) {
            if (this.dict.hasOwnProperty(key)) {
                if (this.isInRectangle(x1, y1, x2, y2, this.dict[key].x, this.dict[key].y)){
                    console.log(this.dict[key]);
                }
            }
        }
    }
}
```

4 Тест программы(Task1)

4.1 Задание 1:

Вывод в конзол:

```
Children in Repository
Olderst child:
```

```
Child: petrovich Age: 15
Child: Chaushev Age: 20
Child: Ivchenko Age: 19
Avarage age: 18
Childer in range from 18 - 25:
Child: Chaushev Age: 20
Child: Ivchenko Age: 19
First Symbol C:
Child: Chaushev Age: 20
Name Longer Than 8:
Child: petrovich Age: 15
Starts on Vowel
Child: Ivchenko Age: 19
```

4.2 Задание 2

Вывод в консол:

```
Students:
Student { group: 'IU6', number: '1112', grades: [ 5, 5, 5, 5 ] }
Student { group: 'IU7', number: '1728', grades: [] }
Student { group: 'IU7', number: '1872', grades: [ 5, 5, 3, 5 ] }
Student { group: 'IU9', number: '1922', grades: [ 3, 3, 3, 5 ] }
Get Avarage Grade Of Student by number:
3.5
Get information of students by group
Student { group: 'IU7', number: '1728', grades: [] }
Student { group: 'IU7', number: '1872', grades: [ 5, 5, 3, 5 ] }
Find Student with the most grades in a group
Student { group: 'IU7', number: '1872', grades: [ 5, 5, 3, 5 ] }
Find StudentWithout grades
Student { group: 'IU7', number: '1728', grades: [] }
```

4.3 Задание 3

Вывод в консол:

```
ALL POINTS
first:
Point { x: 1, y: 6 }
second:
Point { x: 6, y: 2 }
third:
Point { x: -5, y: 12 }
fourth:
Point { x: 0, y: -6 }
```

```
Points at range 11
Point { x: 1, y: 6 }
Point { x: 6, y: 2 }
Point { x: −5, y: 12 }
Points above OX
Point { x: 1, y: 6 }
Point { x: 6, y: 2 }
Point { x: −5, y: 12 }
Points under OX
Point { x: 0, y: −6 }
Points left from Y
Point { x: −5, y: 12 }
Points Right from Y
Point { x: 1, y: 6 }
Point { x: 6, y: 2 }
Find points in rectangle −4 8 3 −5
Point { x: 0, y: −6 }
```

5 Задания (Task 2)

Задание 1

Создать класс Точка. Добавить классу точка Точка метод инициализации полей и метод вывода полей на экран Создать класс Отрезок. У класса Отрезок должны быть поля, являющиеся экземплярами класса Точка. Добавить классу Отрезок метод инициализации полей, метод вывода информации о полях на экран, а так же метод получения длины отрезка.

Задание 2 Создать класс Треугольник. Класс Треугольник должен иметь поля, хранящие длины сторон треугольника.

Реализовать следующие методы:

1. Метод инициализации полей
2. Метод проверки возможности существования треугольника с такими сторонами
3. Метод получения периметра треугольника
4. Метод получения площади треугольника
5. Метод для проверки факта: является ли треугольник прямоугольным

Задание 3 Реализовать программу, в которой происходят следующие действия:

1. Происходит вывод целых чисел от 1 до 10 с задержками в 2 секунды.
2. После этого происходит вывод от 11 до 20 с задержками в 1 секунду.

3. Потом опять происходит вывод чисел от 1 до 10 с задержками в 2 секунды.

4. После этого происходит вывод от 11 до 20 с задержками в 1 секунду.

Это должно происходить циклически.

6 Листинг (Task 2)

6.1 Задание 1

```
    "use strict"

class Point {
    constructor(x, y) {
        this.x = x;
        this.y = y;
    }

    setX(x) {
        this.x = x;
    }

    setY(y) {
        this.y = y;
    }

    showPoint() {
        console.log("X: " + this.x + " Y: " + this.y);
    }
}

class Line {
    constructor(point1, point2) {
        this.point1 = point1;
        this.point2 = point2;
    }

    setPoint1(point) {
        this.point1 = point;
    }

    setPoint2(point) {
        this.point2 = point;
    }

    showLine() {
        console.log("X1: " + this.point1.x + " Y1: " + this.point1.y + "\n" +
        "X2: " + this.point2.x + " Y2: " + this.point2.y)
    }

    getLen() {
        let len = Math.sqrt((this.point1.x - this.point2.x) * (this.point1.x - this.point2.x) +
        (this.point1.y - this.point2.y) * (this.point1.y - this.point2.y));

        return len;
    }
}
```

6.2 Задание 2

```
    "use strict"

class Triangle {
    constructor(a, b, c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    setA(a) {
        this.a = a;
    }

    setB(b) {
        this.b = b;
    }
```

```
    setC(c) {
        this.c = c;
    }

    isCorrect() {
        if (this.a + this.b < this.c) {
            return false;
        }
        if (this.b + this.c < this.a) {
            return false;
        }
        if (this.a + this.c < this.b) {
            return false;
        }

        return true;
    }

    getPerimeter() {
        return this.a + this.b + this.c;
    }

    getS() {
        let p = this.getPerimeter() / 2;
        let s = Math.sqrt(p * (p - this.a) * (p - this.b) * (p - this.c));
        return s;
    }

    findBiggest() {
        let temp = null
        if (this.a >= this.b) {
            temp = this.a;
        } else {
            temp = this.b;
        }
        if (this.c > temp) {
            temp = this.c;
        }
        return temp;
    }

    isPryqmougolnyi() {
        let temp = this.findBiggest();
        if (temp == this.a) {
            if (this.a * this.a == ((this.c * this.c) + (this.b * this.b))) {
                return true;
            }
        }
        if (temp == this.b) {
            if (this.b * this.b  == ((this.a * this.a) + (this.c * this.c))){
                return true;
            }
        }
        if (temp == this.c) {
            if (this.c * this.c == ((this.a * this.a) + (this.b * this.b))){
                return true;
            }
        }
        return false;
    }
}
```

6.3 Задание 3

```
"use strict"

//Outputs from 1 - 10;
function firstOutput() {
    setTimeout(function () {
        for (let i = 1; i <= 10; i++) {
            console.log(i);
        }
    }, 200)
}

//Outputs from 11 - 20
function secondOutput() {
    setTimeout(function() {
        for (let i = 11; i <= 20; i++) {
            console.log(i);
        }
    }, 1000)
}


function timeout() {
    setTimeout(function () {
```

```
        firstOutput()
        secondOutput()
        timeout()
    }, 1000);
}

timeout()
```

7 Тест программы(Task2)

7.1 Задание 1

```
Make 2 Points
Point { x: 0, y: 5 }
Point { x: 5, y: 5 }
Make Line
X1: 0 Y1: 5
X2: 5 Y2: 5
5
```

7.2 Задание 2

```
Create triangle
Triangle { a: 3, b: 4, c: 5 }
perimeter: 12
square: 6
Is Correct:true
IS Pryamougolyni:true
```

7.3 Задание 3

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

```
19
20
1
^ C
```

8 Заключение

8.1 Вывод

После выполнения лабораторной работы я ознакомился с Node.js, Научился работать с целыми числами, Циклы, Массивы, Строки, Объекты, Ссылочний типы данных, Функции в Node.js.