**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»

# Лабораторная работа № 1

Знакомство с синтаксисом языка Java Script

**Студент** Жигалкин Д.Р

**Группа** ИУ7-55Б

**Оценка (баллы) _____**

Москва.
2020 г.

**Цель**: Ознакомиться с синтаксисом языка NodeJS. Изучить особенности работы со строками, объектами, массивами и функциями на данном языке. Научиться создавать и запускать проекты в Visual Studio Code

# *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:

CREATE READ UPDATE DELETE для детей в хранилище

- Получение среднего возраста детей
- Получение информации о самом старшем ребенке
- Получение информации о детях, возраст которых входит в заданный отрезок
- Получение информации о детях, фамилия которых начинается с заданной буквы
- Получение информации о детях, фамилия которых длиннее заданного количества символов
- Получение информации о детях, фамилия которых начинается с гласной буквы

Код:

```
class Child
{
    constructor(secondName, age)
    {
        this.secondName = secondName;
        this.age = age;
    }
}

class ChildStorage
{
    constructor(storage)
    {
        this.storage = storage;
    }

    GetLen()
    {
        let s = storage.length;
        return s;
    }

    Create(child)
    {
```

```
        let id = this.GetIdBySecondName(child.secondName);

        if (id == -1)
        {
            this.storage.push(child);
        }
    }

    GetIdBySecondName(secondName)
    {
        let id = -1;

        for (let i = 0; i < this.storage.length; i++)
        {
            if (this.storage[i].secondName == secondName)
            {
                id = i;
            }
        }

        return id;
    }

    ShowAll()
    {
        for (let i = 0; i < this.storage.length; i++)
        {
            console.log(this.storage[i]);
        }
    }

    Read(id)
    {
        if (id < this.storage.length)
        {
            return this.storage[id];
        }
    }

    Update(id, child)
    {
        let idCheck = this.GetIdBySecondName(child.secondName);

        if (idCheck == -1 || id == idCheck)
        {
            this.storage[id].secondName = child.secondName;
            this.storage[id].age = child.age;
        }
    }

    Delete(id)
```

```
    {
        let answer = null;

        if (id < this.storage.length && id >= 0)
        {
            this.storage.splice(id, 1); // Начиная с позиции id удаляем 1 элемент
            answer = 1;
        }

        return answer;
    }

    GetAverageAge()
    {
        let sum = 0;
        for (let i = 0; i < this.storage.length; i++)
        {
            sum += this.storage[i].age;
        }

        return sum / this.storage.length;
    }

    GetInfoByMaxAge()
    {
        if (this.storage.length == 0)
            return null;

        let maxAgechild = this.storage[0];

        for (let i = 0; i < this.storage.length; i++)
        {
            if (this.storage[i].age > maxAgechild.age)
            {
                maxAgechild = this.storage[i];
            }

        }
        return maxAgechild
    }

    GetChildrenByAge(minAge, maxAge)
    {
        return this.storage.filter(child => minAge <= child.age && child.age <= m
axAge);
    }

    GetChildrenBySecondName(string)
    {
        return this.storage.filter(s => s.secondName.indexOf(string) != -1);
    }
```

```
    GetChildrenByLenSecondName(number)
    {
        return this.storage.filter(s => number < s.secondName.length);
    }

    GetChildrenByVowelSecondName()
    {
        return this.storage.filter(s => s.secondName.search(/^[euioayEYUIOA]/i) !
= -1);
    }
}

let myStorage = new ChildStorage([]);

myStorage.Create(new Child("Afimin", 12));
myStorage.Create(new Child("gf", 1));
myStorage.Create(new Child("A", 2));
myStorage.Create(new Child("Afim", 30));

console.log("show all");
myStorage.ShowAll();
console.log("update id = 2");

myStorage.Update(2, new Child("Afiming", 22));
console.log("show all");
myStorage.ShowAll();
console.log("read[2] = ", myStorage.Read(2));
console.log("delete[2] = ", myStorage.Delete(2));
console.log("show all");
myStorage.ShowAll();

console.log("Read[0]=", myStorage.Read(0));
console.log();

console.log("GetInfoByMaxAge ", myStorage.GetInfoByMaxAge());
console.log("GetChildrenByAge ", myStorage.GetChildrenByAge(1, 2));

console.log("GetChildrenBySecondName ", myStorage.GetChildrenBySecondName("gf"));


console.log("GetChildrenByLenSecondName ", myStorage.GetChildrenByLenSecondName(2
));

console.log("GetChildrenByVowelSecondName ", myStorage.GetChildrenByVowelSecondNa
me());
```

## 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:

- CREATE READ UPDATE DELETE для студентов в хранилище
- Получение средней оценки заданного студента
- Получение информации о студентах в заданной группе
- Получение студента, у которого наибольшее количество оценок в заданной группе
- Получение студента, у которого нет оценок

Код:

```
"use strict";

class Student
{
    constructor(groupName, numberTicket, programmingRate)
    {
        this.groupName = groupName;
        this.numberTicket = numberTicket;
        this.programmingRate = programmingRate;
    }
}

class StudentStorage
{
    constructor(storage)
    {
        this.storage = storage;
    }

    GetLen()
    {
        let s = storage.length;
        return s;
    }
```

```
Create(student)
{
    let id = this.GetIdByTicket(student.numberTicket);

    if (id == -1)
    {
        this.storage.push(student);
    }
}

GetIdByTicket(numberTicket)
{
    let id = -1;

    for (let i = 0; i < this.storage.length; i++)
    {
        if (this.storage[i].numberTicket == numberTicket)
        {
            id = i;
        }
    }

    return id;
}

ShowAll()
{
    for (let i = 0; i < this.storage.length; i++)
    {
        console.log(this.storage[i]);
    }
}

Read(id)
{
    if (id < this.storage.length)
    {
        return this.storage[id];
    }
}

Update(id, student)
{
    let idCheck = this.GetIdByTicket(student.numberTicket);

    if (idCheck == -1 || id == idCheck)
    {
        this.storage[id].groupName = student.groupName;
        this.storage[id].numberTicket = student.numberTicket;
        this.storage[id].programmingRate = student.programmingRate;
```

```
        }
    }

    Delete(id)
    {
        let answer = null;

        if (id < this.storage.length && id >= 0)
        {
            this.storage.splice(id, 1); // Начиная с позиции id удаляем 1 элемент
            answer = 1;
        }

        return answer;
    }

    GetAverageRateStudent(id)
    {
        let sum = 0;
        for (let i = 0; i < this.storage[id].programmingRate.length; i++)
        {
            sum += this.storage[id].programmingRate[i];
        }

        return sum / this.storage[id].programmingRate.length;
    }

    GetInfoByGroup(groupName)
    {
        for (let i = 0; i < this.storage.length; i++)
        {
            if (this.storage[i].groupName == groupName)
            {
                console.log(this.storage[i]);
            }
        }
    }

    GetStudentMaxNumberRate(groupName)
    {
        let answer = null;
        let maxRatenumber = 0;

        for (let i = 0; i < this.storage.length; i++)
        {
            if (this.storage[i].groupName == groupName && this.storage[i].program
mingRate.length > maxRatenumber)
            {
                maxRatenumber = this.storage[i].programmingRate.length;
                answer = this.storage[i];
            }
```

```
        }

        return answer;
    }

    GetStudentZeroRate()
    {
        return this.storage.filter(s => s.programmingRate.length == 0);
    }

}

let myStorage = new StudentStorage([]);

myStorage.Create(new Student("IU7-55", 228, [5, 5, 5]));
myStorage.Create(new Student("IU7-54", 123, [5, 4, 5, 5]));
myStorage.Create(new Student("IU7-54", 113, [2, 2, 5]));
myStorage.Create(new Student("IU7-53", 24, [4, 4, 5, 5, 5, 5]));
myStorage.Create(new Student("IU7-52", 2, [3, 4, 5]));
myStorage.Create(new Student("IU7-51", 23, []));

console.log("show all");
myStorage.ShowAll();

console.log("GetAverageRateStudent id 2 = ", myStorage.GetAverageRateStudent(2));

console.log("GetInfoByGroup IU7-54");
console.log();
myStorage.GetInfoByGroup("IU7-54");

console.log();

console.log("GetStudentMaxNumberRate IU7-
54", myStorage.GetStudentMaxNumberRate("IU7-54"));

console.log();

console.log("GetStudentZeroRate ", myStorage.GetStudentZeroRate());

console.log();
```

## Задание 3

Создать хранилище в оперативной памяти для хранения точек.

Необходимо хранить информацию о точке: имя точки, позиция X и позиция Y.

Необходимо обеспечить уникальность имен точек.

Реализовать функции:

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

Код:

```
"use strict";

class Point
{
    constructor(name, positionX, positionY)
    {
        this.name = name;
        this.positionX = positionX;
        this.positionY = positionY;
    }
}

class PointStorage
{
    constructor(storage)
    {
        this.storage = storage;
    }

    Create(point)
    {
        let id = this.GetIdByName(point.name);

        if (id == -1)
        {
            this.storage.push(point);
        }
    }

    GetIdByName(name)
    {
        let id = -1;

        for (let i = 0; i < this.storage.length; i++)
        {
```

```
            if (this.storage[i].name == name)
            {
                id = i;
            }
        }

        return id;
    }

    ShowAll()
    {
        for (let i = 0; i < this.storage.length; i++)
        {
            console.log(this.storage[i]);
        }
    }

    Read(id)
    {
        if (id < this.storage.length)
        {
            return this.storage[id];
        }
    }

    Update(id, student)
    {
        let idCheck = this.GetIdByName(point.name);

        if (idCheck == -1 || id == idCheck)
        {
            this.storage[id].name = point.name;
            this.storage[id].positionX = point.positionX;
            this.storage[id].positionY = point.positionY;
        }
    }

    Delete(id)
    {
        let answer = null;

        if (id < this.storage.length && id >= 0)
        {
            this.storage.splice(id, 1); // Начиная с позиции id удаляем 1 элемент
            answer = 1;
        }

        return answer;
    }

    GetPointsWithMaxDistance()
```

```
    {
        let answer = [];

        if (this.storage.length <= 1)
            return answer;

        let maxDistance = 0;
        let distance = 0;

        for (let i = 0; i < this.storage.length; i++)
        {
            for (let j = 0; j < this.storage.length; j++)
            {
                distance = this.GetDistance(this.storage[i], this.storage[j]);

                if (distance > maxDistance)
                {
                    answer = [];
                    maxDistance = distance;
                    answer.push(this.storage[i]);
                    answer.push(this.storage[j]);
                }
            }
        }

        return answer;
    }

    GetDistance(firstPoint, secondPoint)
    {
        return Math.sqrt(Math.pow((secondPoint.positionX -
firstPoint.positionX), 2) +
                         Math.pow((secondPoint.positionY -
firstPoint.positionY), 2));
    }

    GetPointCertainDistance(maxDistance, point)
    {
        let answer = [];

        for (let i = 0; i < this.storage.length; i++)
        {
            let distance = this.GetDistance(this.storage[i], point);

            if (distance <= maxDistance)
            {
                answer.push(this.storage[i]);
            }
        }

        return answer;
```

```
}

GetPointsAlongAxis(nameAxis, direction)
{
    let answer = [];

    if (nameAxis == "X")
    {
        for (let i = 0; i < this.storage.length; i++)
        {
            if (this.CheckOnX(this.storage[i]))
            {
                if (direction == "UP")
                {
                    answer.push(this.storage[i]);
                }
            }
            else
            {
                if (direction == "DOWN")
                {
                    answer.push(this.storage[i]);
                }
            }
        }
    }

    if (nameAxis == "Y")
    {
        for (let i = 0; i < this.storage.length; i++)
        {
            if (this.CheckOnY(this.storage[i]))
            {
                if (direction == "RIGHT")
                {
                    answer.push(this.storage[i]);
                }
            }
            else
            {
                if (direction == "LEFT")
                {
                    answer.push(this.storage[i]);
                }
            }
        }
    }

    return answer;
}
```

```
    CheckOnX(point)
    {
        return point.positionY < 0 ? 0 : 1;;
    }

    CheckOnY(point)
    {
        return point.positionX < 0 ? 0 : 1;
    }

    // A(x,y) и C(x1,y1) Точка A является верхней левой вершиной, а C -
 нижней правой.
    GetPointsInZone(firstZonepoint, secondZonepoint)
    {
        let answer = [];
        for (let i = 0; i < this.storage.length; i++)
        {
            if (this.CheckZonePoint(this.storage[i], firstZonepoint, secondZonepo
int))
            {
                answer.push(this.storage[i]);
            }
        }
        return answer;
    }

    CheckZonePoint(point, firstZonepoint, secondZonepoint)
    {
        let answer = 0;

        if (point.positionX >= firstZonepoint.positionX && point.positionX <= sec
ondZonepoint.positionX)
        {
            if (point.positionY <= firstZonepoint.positionY && point.positionY >=
 secondZonepoint.positionY)
            {
                answer = 1;
            }
        }

        return answer;
    }
}

let myStorage = new PointStorage([]);

myStorage.Create(new Point("my", 2, -3));
myStorage.Create(new Point("good", -5, 23));
myStorage.Create(new Point("sad", -10, -23));
myStorage.Create(new Point("bad", -1, 0));
myStorage.Create(new Point("happy", 1, 1));
```

```
myStorage.Create(new Point("invisible", 20, 20));

console.log("show all");
myStorage.ShowAll();

console.log("GetPointsWithMaxDistance ", myStorage.GetPointsWithMaxDistance());

console.log();

console.log("GetPointCertainDistance (100, new Point('test', 0, 0)) = ",
                    myStorage.GetPointCertainDistance(100, new Point("test", 0, 0
)));
console.log("GetPointCertainDistance(2, new Point('test', 0, 0)) = ",
                    myStorage.GetPointCertainDistance(2, new Point("test", 0, 0))
);

console.log();

console.log("GetPointsAlongAxis('X', 'UP') = ", myStorage.GetPointsAlongAxis("X",
 "UP"));
console.log();

console.log("GetPointsAlongAxis(X, DOWN) = ", myStorage.GetPointsAlongAxis("X", "
DOWN"));
console.log();

console.log("GetPointsAlongAxis(Y, RIGHT) = ", myStorage.GetPointsAlongAxis("Y",
"RIGHT"));
console.log();


console.log("GetPointsInZone(new Point(test1, 0, 3), new Point(test2, 2, 0)) = ",

                     myStorage.GetPointsInZone(new Point("test1", 0, 3), new Poin
t("test2", 2, 0)));

console.log();
```

# *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.

```
"use strict";

class Point
{
    constructor(name, positionX, positionY)
    {
        this.name = name;
        this.positionX = positionX;
        this.positionY = positionY;
    }

    ShowInfo()
    {
        console.log("Name: " + this.name + "\nPosition x: " + this.positionX + "\
nPosition y: " + this.positionY);
    }
}

class Section
{
    constructor(firstPoint, secondPoint)
    {
        this.firstPoint = firstPoint;
        this.secondPoint = secondPoint;
    }

    ShowInfo()
    {
        console.log("Info about section\n")
        console.log("First Point:" + "\nPosition x: " + this.firstPoint.positionX
 + "\nPosition y: " + this.firstPoint.positionY);
        console.log("\nSecond Point:" + "\nPosition x: " + this.secondPoint.posit
ionX + "\nPosition y: " + this.secondPoint.positionY);
    }

    GetLength()
    {
        return Math.sqrt(Math.pow((this.secondPoint.positionX -
 this.firstPoint.positionX), 2) +
                         Math.pow((this.secondPoint.positionY -
 this.firstPoint.positionY), 2));
    }
}
```

```
let myPoint = new Point("my", 1, 2);
let myNewpoint = new Point("good", 4, 56);

myPoint.ShowInfo();

console.log();

let section = new Section(myPoint, myNewpoint);

section.ShowInfo();

console.log("\nLength section is " + section.GetLength());
```

## Задание 2

Создать класс *Треугольник*.

Класс *Треугольник* должен иметь поля, хранящие длины сторон треугольника.

Реализовать следующие методы:

- Метод инициализации полей
- Метод проверки возможности существования треугольника с такими сторонами
- Метод получения периметра треугольника
- Метод получения площади треугольника
- Метод для проверки факта: является ли треугольник прямоугольным

```
"use strict";

class Triangle
{
    constructor(lengthA, lengthB, lengthC)
    {
        this.lengthA = lengthA;
        this.lengthB = lengthB;
        this.lengthC = lengthC;
    }

    CheckTriangle()
    {
        let eps = 1e-9;

```

```
        if (this.lengthA + this.lengthB > this.lengthC + eps && this.lengthA + this.lengthC > this.lengthB + eps &&
            this.lengthB + this.lengthC > this.lengthA + eps)
        {
            console.log("It's triangle exist");
            return 1;
        }
        else
        {
            console.log("it's triangle doesn't exist");
            return 0;

        }
    }

    GetPerimetr()
    {
        if (CheckTriangle)
            return this.lengthA + this.lengthB + this.lengthC;
    }

    GetSquare()
    {
        if (CheckTriangle)
        {
            let p = (this.lengthA + this.lengthB + this.lengthC) / 2;
            return Math.sqrt(p * (p - this.lengthA) * (p - this.lengthB) * (p - this.lengthC));
        }
    }

    CheckSquareness()
    {
        if (CheckTriangle)
        {
            let eps = 1e-9;

            let cosA = (this.lengthA * this.lengthA + this.lengthC * this.lengthC - this.lengthB * this.lengthB) /
            2 * this.lengthA * this.lengthC;

            let cosB = (this.lengthA * this.lengthA + this.lengthB * this.lengthB - this.lengthC * this.lengthC) /
            2 * this.lengthA * this.lengthB;

            let cosC = (this.lengthB * this.lengthB + this.lengthC * this.lengthC - this.lengthA * this.lengthA) /
            2 * this.lengthB * this.lengthC;

            if (Math.abs(cosA) < eps || Math.abs(cosB) < eps || Math.abs(cosC) < eps)
```

```
                {
                    console.log("Rectangular triangle");
                }
                else
                {
                    console.log("Not rectangular triangle");
                }
            }

        }
    }

    let myTriangle = new Triangle(2, 4, 5);
    let myTriangle1 = new Triangle(20, 4, 5);
    let myTriangle2 = new Triangle(3, 4, 5);

    myTriangle.CheckTriangle();
    myTriangle1.CheckTriangle();

    console.log(myTriangle.GetPerimetr());
    console.log(myTriangle1.GetPerimetr());

    console.log(myTriangle.GetSquare());
    console.log(myTriangle1.GetSquare());

    myTriangle.CheckSquareness();
    myTriangle1.CheckSquareness();
    console.
    myTriangle2.CheckSquareness();
```

## Задание 3

Реализовать программу, в которой происходят следующие действия:

Происходит вывод целых чисел от 1 до 10 с задержками в 2 секунды.

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

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

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

Это должно происходить циклически.

```
"use strict";

let seconds = 0;

function FirstStep()
{
    let interval = setInterval(() => {
        seconds++;
        let message = "Seconds: " + seconds;
        console.log(message);
        if (seconds === 10)
        {
            clearInterval(interval);
            SecondStep();
        }
    }, 2000);
}

function SecondStep()
{
    let newinterval = setInterval(() => {
        seconds++;
        let message = "Seconds: " + seconds;
        console.log(message);
        if (seconds === 20)
        {
            clearInterval(newinterval);
            seconds = 0;
            FirstStep();
        }
    }, 1000);
}

FirstStep();
```

Вывод: я познакомился с синтаксисом языка JavaScript и изучил особенности работы со строками, объектами, массивами и функциями на данном языке.