|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**По курсу «Архитектура ЭВМ»**

|  |  |
| --- | --- |
| **Тема**  Основы JavaScript  **Студент** Якуба Д. В.  **Группа** ИУ7-53Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Попов А. Ю. |  |

Москва

2020 г.

# Цели работы

* Освоение работы с целыми числами, циклами, строками, массивами, объектами, ссылочными типами данных, функциями и преобразованием в ЯП JavaScript.
* Изучение процедурных типов параметров, области видимости для переменных, основы ООП в ЯП Javascript.
* Освоение использования setTimeout и setInterval в ЯП JavaScript.

# Отчёт по разделу №1

## Задание 1

### Условие

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 программы

Язык: JavaScript

**task1.js**

"use strict";

class KidsData

{

    constructor()

    {

        this.kidsList = [];

    }

    printOut()

    {

        console.log(this.kidsList);

    }

    add(lastName, age)

    {

        let newRecord = {lastName, age};

        if ((this.kidsList.find(kid => kid.lastName === newRecord.lastName)) === undefined)

            this.kidsList.push(newRecord);

    }

    read(lastName)

    {

        return this.kidsList.find(kid => kid.lastName === lastName) || null;

    }

    upd(lastName, age)

    {

        let updateKid = this.read(lastName);

        if (updateKid !== null)

            updateKid.age = age;

    }

    del(delLastName)

    {

        this.kidsList = this.kidsList.filter(curKid => curKid.lastName != delLastName);

    }

    getAverAge()

    {

        if (this.kidsList.length === 0)

            return;

        let summary = 0;

        for (let i = 0; i < this.kidsList.length; i++)

            summary += this.kidsList[i].age;

        return summary / this.kidsList.length;

    }

    getOlderKidInfo()

    {

        if (this.kidsList.length === 0)

            return;

        if (this.kidsList.length === 1)

            return this.kidsList[0].age;

        let olderKid = this.kidsList[0];

        for (let i = 1; i < this.kidsList.length; i++)

            if (this.kidsList[i].age > olderKid.age)

                olderKid = this.kidsList[i];

        return olderKid.age;

    }

    getKidsInfoByAgeSegment(start, end)

    {

        if (this.kidsList.length === 0)

            return;

        return this.kidsList.filter(curKid => start <= curKid.age && curKid.age <= end);

    }

    getKidsInfoByFirstLetter(letter)

    {

            return this.kidsList.filter(curKid => curKid.lastName[0] === letter);

    }

    getKidsInfoWithLongerLastThan(numOfLetters)

    {

        return this.kidsList.filter(curKid => curKid.lastName.length > numOfLetters);

    }

    getKidsInfoLastnStartsWithVowel()

    {

        let vowelList = ['a', 'e', 'i', 'o', 'u'];

        return this.kidsList.filter(curKid => vowelList.find(vowel => vowel === curKid.lastName.toLowerCase()[0]));

    }

};

function main()

{

    let testKids = new KidsData();

    testKids.add("Stalin", 13);

    testKids.add("Lenin", 15);

    testKids.add("Tarasova", 16);

    testKids.add("SCP-1337", 10);

    testKids.add("Somebodyelse", 17);

    console.log("Current kidsData is:")

    testKids.printOut();

    console.log("let's read SCP-1337:\n")

    console.log(testKids.read("SCP-1337"));

    console.log("\nlet's update SCP-1337 and Lenin with new ages:\n");

    console.log("Before:");

    console.log(testKids.read("SCP-1337"), testKids.read("Lenin"));

    testKids.upd("SCP-1337", 5);

    testKids.upd("Lenin", 8);

    console.log("\nAfter:");

    console.log(testKids.read("SCP-1337"), testKids.read("Lenin"));

    console.log("\nlet's remove Stalin!");

    console.log("Before:");

    testKids.printOut();

    testKids.del("Stalin");

    console.log("\nAfter:");

    testKids.printOut();

    console.log("\naverage age of current KidsList is:", testKids.getAverAge());

    console.log("\noldest kiddo is:", testKids.getOlderKidInfo());

    console.log("\nkids of age [6, 17] are:", testKids.getKidsInfoByAgeSegment(6, 17));

    console.log("\nkids of age [5, 15] are:", testKids.getKidsInfoByAgeSegment(5, 15));

    console.log("\nkids lastnames starts with S: ", testKids.getKidsInfoByFirstLetter('S'));

    console.log("\nkids lastnames starts with 5:", testKids.getKidsInfoByFirstLetter('5'));

    console.log("\nkids lastnames longer than 5", testKids.getKidsInfoWithLongerLastThan(5));

    console.log("\nkids lastnames longer than 8", testKids.getKidsInfoWithLongerLastThan(8));

    console.log("\nAdding kids with vowels:");

    testKids.add("Uno", 11);

    testKids.add("Alexaxaxaxaxa", 10);

#### testKids.printOut();

    console.log("\nkids with lastnames starts with vowels:", testKids.getKidsInfoLastnStartsWithVowel());

}

main();

### Результаты тестирования

#### node .\task1.js

#### Debugger attached.

#### Current kidsData is:

#### [

#### { lastName: 'Stalin', age: 13 },

#### { lastName: 'Lenin', age: 15 },

#### { lastName: 'Tarasova', age: 16 },

#### { lastName: 'SCP-1337', age: 10 },

#### { lastName: 'Somebodyelse', age: 17 }

#### ]

#### let's read SCP-1337:

#### { lastName: 'SCP-1337', age: 10 }

#### let's update SCP-1337 and Lenin with new ages:

#### Before:

#### { lastName: 'SCP-1337', age: 10 } { lastName: 'Lenin', age: 15 }

#### After:

#### { lastName: 'SCP-1337', age: 5 } { lastName: 'Lenin', age: 8 }

#### let's remove Stalin!

#### Before:

#### [

#### { lastName: 'Stalin', age: 13 },

#### { lastName: 'Lenin', age: 8 },

#### { lastName: 'Tarasova', age: 16 },

#### { lastName: 'SCP-1337', age: 5 },

#### { lastName: 'Somebodyelse', age: 17 }

#### ]

#### After:

#### [

#### { lastName: 'Lenin', age: 8 },

#### { lastName: 'Tarasova', age: 16 },

#### { lastName: 'SCP-1337', age: 5 },

#### { lastName: 'Somebodyelse', age: 17 }

#### ]

#### average age of current KidsList is: 11.5

#### oldest kiddo is: 17

#### kids of age [6, 17] are: [

#### { lastName: 'Lenin', age: 8 },

#### { lastName: 'Tarasova', age: 16 },

#### { lastName: 'Somebodyelse', age: 17 }

#### ]

#### kids of age [5, 15] are: [ { lastName: 'Lenin', age: 8 }, { lastName: 'SCP-1337', age: 5 } ]

#### kids lastnames starts with S: [

#### { lastName: 'SCP-1337', age: 5 },

#### { lastName: 'Somebodyelse', age: 17 }

#### ]

#### kids lastnames starts with 5: []

#### kids lastnames longer than 5 [

#### { lastName: 'Tarasova', age: 16 },

#### { lastName: 'SCP-1337', age: 5 },

#### { lastName: 'Somebodyelse', age: 17 }

#### ]

#### kids lastnames longer than 8 [ { lastName: 'Somebodyelse', age: 17 } ]

#### Adding kids with vowels:

#### [

#### { lastName: 'Lenin', age: 8 },

#### { lastName: 'Tarasova', age: 16 },

#### { lastName: 'SCP-1337', age: 5 },

#### { lastName: 'Somebodyelse', age: 17 },

#### { lastName: 'Uno', age: 11 },

#### { lastName: 'Alexaxaxaxaxa', age: 10 }

#### ]

#### kids with lastnames starts with vowels: [

#### { lastName: 'Uno', age: 11 },

#### { lastName: 'Alexaxaxaxaxa', age: 10 }

#### ]

## Задание 2

### Условие

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 программы

Язык: JavaScript

**task2.js**

"use strict";

class StudentsData

{

    constructor()

    {

        this.studentsList = [];

    }

    printOut()

    {

        console.log(this.studentsList);

    }

    add(group, studCardNum, progMarks)

    {

        let newRecord = {group, studCardNum, progMarks};

        if ((this.studentsList.find(student => student.studCardNum === newRecord.studCardNum)) === undefined)

            this.studentsList.push(newRecord);

    }

    read(studCardNum)

    {

        return this.studentsList.find(student => student.studCardNum === studCardNum) || null;

    }

    updGroup(studCardNum, group)

    {

        let updateStudent = this.read(studCardNum);

        if (updateStudent !== null)

            updateStudent.group = group;

    }

    updProgMarks(studCardNum, marks)

    {

        let updateStudent = this.read(studCardNum);

        if (updateStudent !== null)

            updateStudent.progMarks = marks;

    }

    upd(studCardNum, group, marks)

    {

        let updateStudent = this.read(studCardNum);

        if (updateStudent !== null)

        {

            updateStudent.group = group;

            updateStudent.progMarks = marks;

        }

    }

    del(delStudCardNum)

    {

        this.studentsList = this.studentsList.filter(curStudent => curStudent.studCardNum !== delStudCardNum);

    }

    getAverageProgMarks(studCardNum)

    {

        if (this.studentsList.length === 0)

            return;

        let student = this.read(studCardNum);

        if (student === null || student.progMarks.length === 0)

            return;

        let summary = 0;

        for (let i = 0; i < student.progMarks.length; i++)

            summary += student.progMarks[i];

        return summary / student.progMarks.length;

    }

    getStudentsByGroup(group)

    {

        return this.studentsList.filter(student => student.group === group);

    }

    getMarkedStudentByGroup(group)

    {

        let students = this.getStudentsByGroup(group);

        if (students.length === 0)

            return;

        let retStud = students[0];

        for (let i = 1; i < students.length; i++)

            if (students[i].progMarks.length > retStud.progMarks.length)

                retStud = students[i];

        return retStud;

    }

    getStudentsWithNoMarks()

    {

        return this.studentsList.filter(student => student.progMarks.length === 0);

    }

};

function main()

{

    let students = new StudentsData();

    students.add("G1", 111, [2, 2, 2]);

    students.add("G1", 112, [2, 5, 3, 5]);

    students.add("G1", 113, [2]);

    students.add("G1", 114, []);

    students.add("G2", 121, [5, 5, 5]);

    students.add("G2", 122, [4, 5, 4, 5 ,5]);

    students.add("G2", 123, [5, 4]);

    students.add("...", 666, []);

    console.log("Starts with data:");

    students.printOut();

    console.log("\nReading 113:", students.read(113));

    console.log("\nDeleting 666:");

    students.del(666);

    students.printOut();

    console.log("\nUpd for 113 with new marks:");

    console.log("Before:", students.read(113));

    students.updProgMarks(113, [3, 3]);

    console.log("\nAfter:", students.read(113));

    console.log("\nUpd for 113 with new group:");

    console.log("Before:", students.read(113));

    students.updGroup(113, "WOHOOOO");

    console.log("\nAfter:", students.read(113));

    console.log("\nBack to start for 113:");

    console.log("Before:", students.read(113));

    students.upd(113, "G1", [2]);

    console.log("After:", students.read(113));

    console.log("\nAverage in marks for 122", students.read(122), "AVERAGE: ", students.getAverageProgMarks(122));

    console.log("\nAverage in marks for 111", students.read(111), "AVERAGE: ", students.getAverageProgMarks(111));

    console.log("\nStudents of goups: ", students.getStudentsByGroup("G1"), students.getStudentsByGroup("G2"));

    console.log("\nMost marked student of group G1:", students.getMarkedStudentByGroup("G1"));

    console.log("\nMost marked student of group G2:", students.getMarkedStudentByGroup("G2"));

    console.log("\nStudents with no marks:", students.getStudentsWithNoMarks());

}

main();

### Результаты тестирования

#### node .\task2.js

#### Starts with data:

#### [

#### { group: 'G1', studCardNum: 111, progMarks: [ 2, 2, 2 ] },

#### { group: 'G1', studCardNum: 112, progMarks: [ 2, 5, 3, 5 ] },

#### { group: 'G1', studCardNum: 113, progMarks: [ 2 ] },

#### { group: 'G1', studCardNum: 114, progMarks: [] },

#### { group: 'G2', studCardNum: 121, progMarks: [ 5, 5, 5 ] },

#### { group: 'G2', studCardNum: 122, progMarks: [ 4, 5, 4, 5, 5 ] },

#### { group: 'G2', studCardNum: 123, progMarks: [ 5, 4 ] },

#### { group: '...', studCardNum: 666, progMarks: [] }

#### ]

#### Reading 113: { group: 'G1', studCardNum: 113, progMarks: [ 2 ] }

#### Deleting 666:

#### [

#### { group: 'G1', studCardNum: 111, progMarks: [ 2, 2, 2 ] },

#### { group: 'G1', studCardNum: 112, progMarks: [ 2, 5, 3, 5 ] },

#### { group: 'G1', studCardNum: 113, progMarks: [ 2 ] },

#### { group: 'G1', studCardNum: 114, progMarks: [] },

#### { group: 'G2', studCardNum: 121, progMarks: [ 5, 5, 5 ] },

#### { group: 'G2', studCardNum: 122, progMarks: [ 4, 5, 4, 5, 5 ] },

#### { group: 'G2', studCardNum: 123, progMarks: [ 5, 4 ] }

#### ]

#### Upd for 113 with new marks:

#### Before: { group: 'G1', studCardNum: 113, progMarks: [ 2 ] }

#### After: { group: 'G1', studCardNum: 113, progMarks: [ 3, 3 ] }

#### Upd for 113 with new group:

#### Before: { group: 'G1', studCardNum: 113, progMarks: [ 3, 3 ] }

#### After: { group: 'WOHOOOO', studCardNum: 113, progMarks: [ 3, 3 ] }

#### Back to start for 113:

#### Before: { group: 'WOHOOOO', studCardNum: 113, progMarks: [ 3, 3 ] }

#### After: { group: 'G1', studCardNum: 113, progMarks: [ 2 ] }

#### Average in marks for 122 { group: 'G2', studCardNum: 122, progMarks: [ 4, 5, 4, 5, 5 ] } AVERAGE: 4.6

#### Average in marks for 111 { group: 'G1', studCardNum: 111, progMarks: [ 2, 2, 2 ] } AVERAGE: 2

#### Students of goups: [

#### { group: 'G1', studCardNum: 111, progMarks: [ 2, 2, 2 ] },

#### { group: 'G1', studCardNum: 112, progMarks: [ 2, 5, 3, 5 ] },

#### { group: 'G1', studCardNum: 113, progMarks: [ 2 ] },

#### { group: 'G1', studCardNum: 114, progMarks: [] }

#### ] [

#### { group: 'G2', studCardNum: 121, progMarks: [ 5, 5, 5 ] },

#### { group: 'G2', studCardNum: 122, progMarks: [ 4, 5, 4, 5, 5 ] },

#### { group: 'G2', studCardNum: 123, progMarks: [ 5, 4 ] }

#### ]

#### Most marked student of group G1: { group: 'G1', studCardNum: 112, progMarks: [ 2, 5, 3, 5 ] }

#### Most marked student of group G2: { group: 'G2', studCardNum: 122, progMarks: [ 4, 5, 4, 5, 5 ] }

#### Students with no marks: [ { group: 'G1', studCardNum: 114, progMarks: [] } ]

## Задание 3

### Условие

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:

* 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 программы

Язык: JavaScript

**task3.js**

"use strict";

class DotsData

{

    constructor()

    {

        this.dotsList = [];

    }

    printOut()

    {

        console.log(this.dotsList);

    }

    add(name, xPos, yPos)

    {

        let newRecord = {name, xPos, yPos};

        if ((this.dotsList.find(dot => dot.name === newRecord.name)) === undefined)

            this.dotsList.push(newRecord);

    }

    read(name)

    {

        return this.dotsList.find(dot => dot.name === name) || null;

    }

    updX(name, newXPos)

    {

        let updDot = this.read(name);

        if (updDot !== null)

            updDot.xPos = newXPos;

    }

    updY(name, newYPos)

    {

       let updDot = this.read(name);

       if (updDot !== null)

           updDot.yPos = newYPos;

   }

    upd(name, newXPos, newYPos)

    {

        let updDot = this.read(name);

        if (updDot !== null)

       {

           updDot.xPos = newXPos;

           updDot.yPos = newYPos;

       }

    }

    del(delName)

    {

        this.dotsList = this.dotsList.filter(dot => dot.name !== delName);

    }

    getDistance(fDotName, sDotName)

    {

        let fDot = this.read(fDotName);

        let sDot = this.read(sDotName);

        if (fDot === null || sDot === null)

            return;

        let xInc = fDot.xPos - sDot.xPos;

        let yInc = fDot.yPos - sDot.yPos;

        return Math.sqrt(xInc \* xInc + yInc \* yInc);

    }

    getMostDistantDots()

    {

        if (this.dotsList.length < 2)

            return;

        let maxDistDots = [this.dotsList[0], this.dotsList[1]];

        let maxDist = this.getDistance(this.dotsList[0].name, this.dotsList[1].name);

        for (let fDot of this.dotsList)

            for (let sDot of this.dotsList)

            {

                let dist = this.getDistance(fDot.name, sDot.name);

                if (dist > maxDist)

                {

                    maxDistDots[0] = fDot;

                    maxDistDots[1] = sDot;

                    maxDist = dist;

                }

            }

        return maxDistDots;

    }

    getDotsOnDistanceFromDot(mainDotName, distance)

    {

        return this.dotsList.filter(dot => this.getDistance(mainDotName, dot.name) && this.getDistance(mainDotName, dot.name) <= distance)

    }

    getDotsInCoordinateQuarter(axis, position)

    {

        let filterfunc;

        if (axis.toLowerCase() === "x")

        {

            if (position.toLowerCase() === "lower")

                filterfunc = dot => dot.xPos < 0;

            else

                filterfunc = dot => dot.xPos > 0;

        }

        else

        {

            if (position.toLowerCase() === "lower")

                filterfunc = dot => dot.yPos < 0;

            else

                filterfunc = dot => dot.yPos > 0;

        }

        return this.dotsList.filter(filterfunc);

    }

    getDotsInSquare(maxX, maxY, minX, minY)

    {

        return this.dotsList.filter(dot => dot.xPos <= maxX && dot.yPos <= maxY && dot.xPos >= minX && dot.yPos >= minY);

    }

};

function main()

{

    let dots = new DotsData();

    dots.add("a", 10, 20);

    dots.add("b", -10, 20);

    dots.add("c", 10, -20);

    dots.add("d", -10, -23);

    dots.add("so far", 200, 300);

    dots.add("uwu", 666, -888);

    console.log("Current dots:");

    dots.printOut();

    console.log("Read dot d:", dots.read("d"));

    console.log("Delete dot uwu:");

    console.log("Before:");

    dots.printOut();

    dots.del("uwu");

    console.log("\nAfter:");

    dots.printOut();

    console.log("Upd of xPos of so far dot:");

    dots.updX("so far", -11111111);

    dots.printOut();

    console.log("Upd of yPos of so far dot:");

    dots.updY("so far", -2222);

    dots.printOut();

    console.log("Back total upd for so far dot:");

    dots.upd("so far", 200, 300);

    dots.printOut();

    console.log("The most distant dots are:", dots.getMostDistantDots());

    console.log("All dots distanted on 40 from d dot: ", dots.getDotsOnDistanceFromDot("d", 40));

    console.log("All dots distanted on 100 from d dot: ", dots.getDotsOnDistanceFromDot("d", 100));

    console.log("All dots lower x:", dots.getDotsInCoordinateQuarter("x", "lower"));

    console.log("All dots heigher x:", dots.getDotsInCoordinateQuarter("x", "heigher"));

    console.log("All dots lower y:", dots.getDotsInCoordinateQuarter("y", "lower"));

    console.log("All dots heigher y:", dots.getDotsInCoordinateQuarter("y", "heigher"));

    console.log("All dots in square 0, 0, 500, 400", dots.getDotsInSquare(500, 400, 0, 0));

    console.log("All dots in square 100, 100, -100, -100", dots.getDotsInSquare(100, 100, -100, -100));

}

main();

### Результаты тестирования

#### node .\task3.js

#### Current dots:

#### [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 },

#### { name: 'so far', xPos: 200, yPos: 300 },

#### { name: 'uwu', xPos: 666, yPos: -888 }

#### ]

#### Read dot d: { name: 'd', xPos: -10, yPos: -23 }

#### Delete dot uwu:

#### Before:

#### [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 },

#### { name: 'so far', xPos: 200, yPos: 300 },

#### { name: 'uwu', xPos: 666, yPos: -888 }

#### ]

#### After:

#### [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 },

#### { name: 'so far', xPos: 200, yPos: 300 }

#### ]

#### Upd of xPos of so far dot:

#### [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 },

#### { name: 'so far', xPos: -11111111, yPos: 300 }

#### ]

#### Upd of yPos of so far dot:

#### [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 },

#### { name: 'so far', xPos: -11111111, yPos: -2222 }

#### ]

#### Back total upd for so far dot:

#### [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 },

#### { name: 'so far', xPos: 200, yPos: 300 }

#### ]

#### The most distant dots are: [

#### { name: 'd', xPos: -10, yPos: -23 },

#### { name: 'so far', xPos: 200, yPos: 300 }

#### ]

#### All dots distanted on 40 from d dot: [ { name: 'c', xPos: 10, yPos: -20 } ]

#### All dots distanted on 100 from d dot: [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 }

#### ]

#### All dots lower x: [

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'd', xPos: -10, yPos: -23 }

#### ]

#### All dots heigher x: [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'so far', xPos: 200, yPos: 300 }

#### ]

#### All dots lower y: [

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 }

#### ]

#### All dots heigher y: [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'so far', xPos: 200, yPos: 300 }

#### ]

#### All dots in square 0, 0, 500, 400 [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'so far', xPos: 200, yPos: 300 }

#### ]

#### All dots in square 100, 100, -100, -100 [

#### { name: 'a', xPos: 10, yPos: 20 },

#### { name: 'b', xPos: -10, yPos: 20 },

#### { name: 'c', xPos: 10, yPos: -20 },

#### { name: 'd', xPos: -10, yPos: -23 }

#### ]

# Отчёт по разделу №2

## Задание 1

### Условие

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.

### Код программы

Язык: JavaScript

**task1.js**

"use strict";

class Dot

{

    constructor(xPos, yPos)

    {

        this.set(xPos, yPos);

    }

    set(xPos\_, yPos\_)

    {

        this.xPos = xPos\_;

        this.yPos = yPos\_;

    }

    printOut()

    {

        console.log("Dot(", this.xPos, ",", this.yPos, ")");

    }

};

class Section

{

    constructor(fDot, sDot)

    {

        this.set(fDot, sDot);

    }

    set(fDot\_, sDot\_)

    {

        this.fDot = fDot\_;

        this.sDot = sDot\_;

    }

    printOut()

    {

        console.log("Section{ ( ", this.fDot.xPos, this.fDot.yPos, "); (", this.sDot.xPos, this.sDot.yPos, ")");

    }

    getLength()

    {

        let xInc = this.fDot.xPos - this.sDot.xPos;

        let yInc = this.fDot.yPos - this.sDot.yPos;

        return Math.sqrt(xInc \* xInc + yInc \* yInc);

    }

};

function main()

{

    console.log("Init new dot with position 666, 1337: ");

    let showDot = new Dot(666, 1337);

    showDot.printOut();

    console.log("Init new dots for section:");

    let fDot = new Dot(0, 0);

    let sDot = new Dot(10, 10);

    fDot.printOut();

    sDot.printOut();

    console.log("Init section:");

    let showSection = new Section(fDot, sDot);

    showSection.printOut();

    console.log("Section's length is:", showSection.getLength());

}

main();

### Результаты тестирования

#### node .\task01.js

#### Init new dot with position 666, 1337:

#### Dot( 666 , 1337 )

#### Init new dots for section:

#### Dot( 0 , 0 )

#### Dot( 10 , 10 )

#### Init section:

#### Section{ ( 0 0 ); ( 10 10 )

#### Section's length is: 14.142135623730951

## Задание 2

### Условие

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

### Код программы

Язык: JavaScript

**task2.js**

"use strict";

class Triangle

{

    constructor(fSide, sSide, tSide)

    {

        this.set(fSide, sSide, tSide);

    }

    printOut()

    {

        console.log("Triangle: (", this.fSide, ",", this.sSide, ",", this.tSide, ")");

    }

    set(fSide\_, sSide\_, tSide\_)

    {

        this.fSide = fSide\_;

        this.sSide = sSide\_;

        this.tSide = tSide\_;

    }

    get()

    {

        return {fSide: this.fSide, sSide: this.sSide, tSide: this.tSide};

    }

    isPossible()

    {

        let {fSide, sSide, tSide} = this.get();

        return fSide < sSide + tSide && sSide < fSide + tSide && tSide < fSide + sSide && fSide && sSide && tSide;

    }

    getPerimeter()

    {

        if (this.isPossible() === false)

            return;

        let {fSide, sSide, tSide} = this.get();

        return fSide + sSide + tSide;

    }

    getArea()

    {

        if (this.isPossible() === false)

            return;

        let halfPerimiter = this.getPerimeter() / 2;

        let {fSide, sSide, tSide} = this.get();

        return Math.sqrt(halfPerimiter \* (halfPerimiter - fSide) \* (halfPerimiter - sSide) \* (halfPerimiter - tSide));

    }

    checkPythahoras(sqrFSide, sqrSSide, sqrTSide)

    {

        return Math.abs(sqrFSide - sqrSSide - sqrTSide) < 0.0001;

    }

    isRightTriangle()

    {

        if (this.isPossible() === false)

            return;

        let {fSide, sSide, tSide} = this.get();

        fSide \*= fSide;

        sSide \*= sSide;

        tSide \*= tSide;

        return this.checkPythahoras(fSide, sSide, tSide) || this.checkPythahoras(sSide, fSide, tSide) || this.checkPythahoras(tSide, fSide, sSide);

    }

};

function main()

{

    let showTri = new Triangle(10, 10, Math.sqrt(200));

    let impossibleTri = new Triangle(10, 10, 100);

    console.log("Triangle:");

    impossibleTri.printOut();

    console.log("Is this possible? Answer: ", impossibleTri.isPossible());

    console.log("Triangle:");

    showTri.printOut();

    console.log("Is this triangle possible? Answer:", showTri.isPossible());

    console.log("Nice. Let's begin.");

    console.log("Perimeter of this angle is:", showTri.getPerimeter());

    console.log("Area of this triangle is:", showTri.getArea());

    console.log("Is this triangle right? Answer:", showTri.isRightTriangle());

}

main();

### Результаты тестирования

#### node .\task02.js

#### Triangle:

#### Triangle: ( 10 , 10 , 100 )

#### Is this possible? Answer: false

#### Triangle:

#### Triangle: ( 10 , 10 , 14.142135623730951 )

#### Is this triangle possible? Answer: 14.142135623730951

#### Nice. Let's begin.

#### Perimeter of this angle is: 34.14213562373095

#### Area of this triangle is: 50

#### Is this triangle right? Answer: true

## Задание 3

### Условие

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.

### Код программы

Язык: JavaScript

**task3.js**

"use strict";

let firstTime = 2000;

let secondTime = 1000;

let endOfCycles = 2;

let counter = 0;

function sTimer()

{

    counter++;

    console.log(counter);

    if (counter < 20)

        setTimeout(sTimer, secondTime);

    else

    {

        endOfCycles--;

        main();

    }

}

function fTimer()

{

    counter++;

    console.log(counter);

    if (counter < 11)

        setTimeout(fTimer, firstTime);

    else

        setTimeout(sTimer, secondTime);

}

function main()

{

    counter = 0;

    if (endOfCycles > 0)

        fTimer();

}

main();

### Результаты тестирования

#### node .\task03.js

#### 1

#### 2

#### 3

#### 4

#### 5

#### 6

#### 7

#### 8

#### 9

#### 10

#### 11

#### 12

#### 13

#### 14

#### 15

#### 16

#### 17

#### 18

#### 19

#### 20

#### 1

#### 2

#### 3

#### 4

#### 5

#### 6

#### 7

#### 8

#### 9

#### 10

#### 11

#### 12

#### 13

#### 14

#### 15

#### 16

#### 17

#### 18

#### 19

#### 20

# Вывод

В результате выполнения работы:

* Была освоена работа с целыми числами, циклами, строками, массивами, объектами, ссылочными типами данных, функциями и преобразованием в ЯП JavaScript.
* Были изучены процедурные типы параметров, область видимости для переменных, основы ООП в ЯП Javascript.
* Было освоено использование setTimeout и setInterval в ЯП JavaScript.