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 № 1-2

Тема: Знакомство с синтаксисом языка Java Script

Студент Унтилова А.О.

Группа ИУ7-56

Оценка (баллы) _____

Преподаватель Попов А.Ю.

Москва.
2020 г.

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

Лабораторная работа №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

```
"use strict"

class KidsGroup {

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

    create_kid(lastname, age) {
        if (this.kids[lastname]) {
            console.log("Kid with lastname " + lastname + " already exist!");
        }
        this.kids[lastname] = age;
    }

    delete_kid(lastname) {
        if (this.kids[lastname]) {
            delete this.kids[lastname];
        }
    }

    read_kids() {
        for (let i in this.kids) {
            console.log("Kid: " + i + " " + "Age: " + this.kids[i]);
        }
    }

    update_kid_age(lastname, age) {
```

```
        if (this.kids[lastname]) {
            this.kids[lastname] = age
        }
        else {
            console.log("No kid with lastname " + lastname + " !");
        }
    }

    average_age() {
        let sum = 0;
        let num = 0;
        for (let i in this.kids) {
            num++;
            sum += this.kids[i];
        }
        if (num) {
            console.log("Average age = " + (sum / num) );
        } else {
            console.log("There are no kids in group!");
        }
    }

    oldest_kid() {
        let res = null;
        for (let i in this.kids) {
            if ((res == null) || ((this.kids[i] > this.kids[res]) && (res))) {
                res = i;
            }
        }
        console.log("Oldest Kid: " + res + " " + "Age: " + this.kids[res]);
    }

    age_in_range(down, up) {
        for (let i in this.kids) {
            if (this.kids[i] > down && this.kids[i] < up) {
                console.log("Kid: " + i + " " + "Age: " + this.kids[i]);
            }
        }
    }

    first_symbol(sym) {
        for (let i in this.kids) {
            if (i[0] == sym) {
                console.log("Kid: " + i + " " + "Age: " + this.kids[i]);
            }
        }
    }

    len_lastname_longer_than(max_len) {
        for (let i in this.kids) {
            if (i.length > max_len) {
                console.log("Kid: " + i + " " + "Age: " + this.kids[i]);
            }
        }
    }

    first_symbol_vowel() {
        let mas_vowel = "a A e E i I o O u U y Y";
        for (let i in this.kids) {
            if (mas_vowel.includes(i[0])) {
                console.log("Kid: " + i + " " + "Age: " + this.kids[i]);
            }
        }
    }
}
```

```
let group = new KidsGroup();
```

Тесты:

Create, Read:

```
group.create_kid("Petrov", 15);
group.create_kid("Vetrov", 10);
group.create_kid("Ivanova", 4);
group.create_kid("Antipova", 14);
group.create_kid("Sidorov", 6);
group.create_kid("Solovev", 9);
group.create_kid("Galinskaya", 19);

group.read_kids();
```

```
Kid: Petrov Age: 15
Kid: Vetrov Age: 10
Kid: Ivanova Age: 4
Kid: Antipova Age: 14
Kid: Sidorov Age: 6
Kid: Solovev Age: 9
Kid: Galinskaya Age: 19
```

Delete:

```
group.delete_kid("Petrov");
```

```
Kid: Vetrov Age: 10
Kid: Ivanova Age: 4
Kid: Antipova Age: 14
Kid: Sidorov Age: 6
Kid: Solovev Age: 9
Kid: Galinskaya Age: 19
```

Update:

```
group.update_kid_age("Solovev", 11);
```

```
Kid: Vetrov Age: 10
Kid: Ivanova Age: 4
Kid: Antipova Age: 14
Kid: Sidorov Age: 6
Kid: Solovev Age: 11
Kid: Galinskaya Age: 19
```

Получение среднего возраста детей:

```
group.average_age();
```

```
Average age = 10.666666666666666
```

Получение информации о самом старшем ребенке:

```
group.oldest_kid();
```

```
Oldest Kid: Galinskaya Age: 19
```

Получение информации о детях, возраст которых входит в заданный отрезок:

`group.age_in_range(8, 17);`	`Kid: Vetrov Age: 10`
	`Kid: Antipova Age: 14`
	`Kid: Solovev Age: 11`

Получение информации о детях, фамилия которых начинается с заданной буквы:

`group.first_symbol("S");`	`Kid: Sidorov Age: 6` `Kid: Solovev Age: 11`

Получение информации о детях, фамилия которых длиннее заданного количества символов:

`group.len_lastname_longer_than(7);`	`Kid: Antipova Age: 14` `Kid: Galinskaya Age: 19`

Получение информации о детях, фамилия которых начинается с гласной буквы:

`group.first_symbol_vowel();`	`Kid: Ivanova Age: 4` `Kid: Antipova Age: 14`

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

```
class Student {

    constructor(name, num, mas) {
        this.stud = {};
        this.stud["gr_name"] = name;
        this.stud["st_card"] = num;
        this.stud["marks"] = mas;
    }

    average_mark() {
```

```
        let num = 0;
        let sum = 0;
        let mas = this.stud["marks"];
        for(let i = 0; i < (mas.length); i++) {
            num++;
            sum += mas[i];
        }
        if (num) {
            console.log("Average mark = " + (sum / num) );
        } else {
            console.log("There are no marks!");
        }
    }

    read_info(){
        let string = "";
        for (let j in this.stud) {
            string += j + ": " + this.stud[j] + " ";
        }
        console.log(string);
    }
}


function create_stud(group, name, num, mas) {
    let flag = 0;
    for(let i = 0; i < group.length && flag == 0; i++){
        let st = group[i];
        if (st.stud["st_card"] == num){
            flag = 1;
        }
    }
    if (!flag){
        let st = new Student(name, num, mas);
        group.push(st);
    }
    else{
        console.log("Student with student card number: " + num + " already
exists!\n");
    }
}

function delete_stud(group, name) {
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        if (st.stud["gr_name"] == name) {
            group.splice(i, 1);
        }
    }
}

function update_stud_group(group, num, name) {
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        if (st.stud["st_card"] == num) {
            st.stud["gr_name"] = name;
        }
    }
}

function read_stud(group) {
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        st.read_info();
    }
```

```
}

function find_average_mark(group, num){
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        if(st.stud["st_card"] == num){
            st.average_mark();
        }
    }
}

function student_in_group(group, name){
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        if(st.stud["gr_name"] == name){
            st.read_info();
        }
    }
}

function max_num_marks(group, name){
    let res = null;
    let res_st = null;
    if (group.length){}
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        if(st.stud["gr_name"] == name){
            let mas = st.stud["marks"];
            if ((res == null) || ((mas.length > res) && (res))) {
                res = mas.length;
                res_st = group[i];
            }
        }
    }
    if (res_st){
        res_st.read_info();
        console.log("Num of marks: " + res);
    }
}

function no_marks(group){
    let res_st = null;
    if (group.length){}
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        let mas = st.stud["marks"];
        if (mas.length == 0) {
            res_st = group[i];
        }
    }
    if (res_st){
        res_st.read_info();
    }
}

let group = [];
```

Тесты:

Create, Read:

`create_stud(group, "IU7-56", 969, [4, 2, 3]);`	`gr_name: IU7-56 st_card: 969 marks: 4,2,3`

```
create_stud(group, "MT11-12", 910, [5, 3, 4, 3]);
create_stud(group, "IU3-53", 942, [3, 4, 3]);
create_stud(group, "RL5-36", 814, [4, 5, 3]);
create_stud(group, "RK7-56", 863, []);
create_stud(group, "MT4-51", 944, [5, 5, 3]);
create_stud(group, "IU3-53", 819, [5, 5, 3, 4, 5, 5]);
create_stud(group, "IU4-23", 819, [5, 5, 3]);

read_stud(group);
```

```
gr_name: MT11-12 st_card: 910 marks: 5,3,4,3
gr_name: IU3-53 st_card: 942 marks: 3,4,3
gr_name: RL5-36 st_card: 814 marks: 4,5,3
gr_name: RK7-56 st_card: 863 marks:
gr_name: MT4-51 st_card: 944 marks: 5,5,3
gr_name: IU3-53 st_card: 819 marks: 5,5,3,4,5,5
```

Delete:

```
delete_stud(group, "IU7-56");
```

```
gr_name: MT11-12 st_card: 910 marks: 5,3,4,3
gr_name: IU3-53 st_card: 942 marks: 3,4,3
gr_name: IU6-32 st_card: 814 marks: 4,5,3
gr_name: RK7-56 st_card: 863 marks:
gr_name: MT4-51 st_card: 944 marks: 5,5,3
gr_name: IU3-53 st_card: 819 marks: 5,5,3,4,5,5
```

Update:

```
update_stud_group(group, 814, "IU6-32");
```

```
gr_name: IU6-32 st_card: 814 marks: 4,5,3
```

Получение средней оценки заданного студента

```
find_average_mark(group, 819);
```

```
Average mark = 4.5
```

Получение информации о студентах в заданной группе

```
student_in_group(group, "IU3-53");
```

```
gr_name: IU3-53 st_card: 942 marks: 3,4,3
gr_name: IU3-53 st_card: 819 marks: 5,5,3,4,5,5
```

Получение студента, у которого наибольшее количество оценок в заданной группе

```
max_num_marks(group, "IU3-53");
```

```
gr_name: IU3-53 st_card: 819 marks: 5,5,3,4,5,5
Num of marks: 6
```

Получение студента, у которого нет оценок

`no_marks(group);`	`gr_name: RK7-56 st_card: 863 marks:`

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:

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

```
class Point {

    constructor(name, x, y) {
        this.pnt = {};
        this.pnt["name"] = name;
        this.pnt["x_cord"] = x;
        this.pnt["y_cord"] = y;
    }

    read_info(flag = 1){
        let string = "";
        for (let j in this.pnt) {
            string += j + ": " + this.pnt[j] + " ";
        }
        if (flag)
            console.log(string);
        return string;
    }
}


function create_point(group, name, x, y) {
    let flag = 0;
    for(let i = 0; i < group.length && flag == 0; i++){
        let st = group[i];
        if (st.pnt["name"] == name){
            flag = 1;
```

```
        }
    }
    if (!flag){
        let st = new Point(name, x, y);
        group.push(st);
    }
    else{
        console.log("Point with such name: " + name + " already exists!\n");
    }
}

function delete_point(group, name) {
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        if (st.pnt["name"] == name) {
            group.splice(i, 1);
        }
    }
}

function update_point_group(group, name, x) {
    for (let i = 0; i < group.length; i++) {
        let st = group[i];
        if (st.pnt["name"] == name) {
            st.pnt["x_cord"] = x;
        }
    }
}

function read_point(group) {
    for (let i = 0; i < group.length; i++) {
        let pt = group[i];
        pt.read_info();
    }
}

function find_len(x1, y1, x2, y2){
    let x = x2 - x1;
    let y = y2 - y1;
    let dist = Math.sqrt(x * x + y * y);
    return dist;
}

function find_max_dist(group){
    let max_dist = -1, ind1, ind2;
    if (group.length > 1){
        let ind1 = 0;
        let ind2 = 1;
    }
    for (let i = 0; i < group.length - 1; i++) {
        let pt1 = group[i];
        for (let j = i + 1; j < group.length; j++) {
            let pt2 = group[j];
            let dist = find_len(pt1.pnt["x_cord"], pt1.pnt["y_cord"],
pt2.pnt["x_cord"], pt2.pnt["y_cord"]);
            if (max_dist < 0){
                max_dist = dist;
            }
            else if(dist > max_dist){
                max_dist = dist;
                ind1 = i;
                ind2 = j;
            }
        }
    }
```

```
    let ans = "Максимальное расстояние расстояние между двумя точками = " + max_dist
+ " между точками: ";
    console.log(ans);
    ans += group[ind1].read_info();
    ans += group[ind2].read_info();
}


function find_dist_less_than(group, pt, max_dist){
    let str = "Искомые точки: ";
    for (let i = 0; i < group.length ; i++) {
        let pt2 = group[i];
        let dist = find_len(pt.pnt["x_cord"], pt.pnt["y_cord"], pt2.pnt["x_cord"],
pt2.pnt["y_cord"]);
        if (dist <= max_dist){
            str += pt2.pnt["name"] + " ";
        }
    }
    console.log(str);
}

function points_axes(group){
    let pt = null;
    let num = 0;
    let str = "";
    let x1, y1;
    for (let i = 0; i < group.length ; i++) {
        str = "";
        pt = group[i];
        str += pt.pnt["name"];
        x1 = pt.pnt["x_cord"];
        y1 = pt.pnt["y_cord"];
        if(x1 > num){
            str += " правее оси Oy";
        }
        else if(x1 < num){
            str += " левее оси Oy";
        }
        else{
            str += " на оси Oy";
        }

        if(y1 > num){
            str += " выше оси Ox";
        }
        else if(y1 < num){
            str += " ниже оси Ox";
        }
        else{
            str += " на оси Ox";
        }
        console.log(str);
    }
}

function find_points_in_zone(group, xr1, yr1, xr2, yr2){
    let str = " ' заданной области лежат точки: ";
    let pt;
    for (let i = 0; i < group.length; i++){
        pt = group[i];
        let x1 = pt.pnt["x_cord"];
        let y1 = pt.pnt["y_cord"];
        if ((x1 >= xr1 && x1 <= xr2 && y1 <= yr2 && y1 >= yr1) || (x1 >= xr2 && x1 <=
xr1 && y1 <= yr1 && y1 >= yr2)){
```

```
            str += pt.pnt["name"] + " ";
        }
    }
    console.log(str);
}

let group_points = [];
```

Тесты:

Create, Read:

```
create_point(group_points, "A", 3, 2);
create_point(group_points, "B", 14, -3);
create_point(group_points, "C", -2, 6);
create_point(group_points, "D", 0, 10);
create_point(group_points, "K", -5, -3);
create_point(group_points, "T", -1, -2);
create_point(group_points, "M", 1, 1);
create_point(group_points, "N", 4, 5);
read_point(group_points);
```

```
name: A x_cord: 3 y_cord: 2
name: B x_cord: 14 y_cord: -3
name: C x_cord: -2 y_cord: 6
name: D x_cord: 0 y_cord: 10
name: K x_cord: -5 y_cord: -3
name: T x_cord: -1 y_cord: -2
name: M x_cord: 1 y_cord: 1
name: N x_cord: 4 y_cord: 5
```

Delete:

```
delete_point(group_points, "A");
```

```
name: B x_cord: 14 y_cord: -3
name: C x_cord: -2 y_cord: 6
name: D x_cord: 0 y_cord: 10
name: K x_cord: -5 y_cord: -3
name: T x_cord: -1 y_cord: -2
name: M x_cord: 1 y_cord: 1
name: N x_cord: 4 y_cord: 5
```

Update:

```
update_point_group(group_points, "A",
11);
```

```
name: A x_cord: 11 y_cord: 2
name: B x_cord: 14 y_cord: -3
name: C x_cord: -2 y_cord: 6
name: D x_cord: 0 y_cord: 10
name: K x_cord: -5 y_cord: -3
name: T x_cord: -1 y_cord: -2
name: M x_cord: 1 y_cord: 1
name: N x_cord: 4 y_cord: 5
```

Получение двух точек, между которыми наибольшее расстояние

```
find_max_dist(group_points);
```

```
Максимальное расстояние расстояние между
двумя точками = 19.1049731745428 между
точками:
name: B x_cord: 14 y_cord: -3
name: D x_cord: 0 y_cord: 10
```

Получение точек, находящихся от заданной точки на расстоянии, не превышающем заданную константу

`find_dist_less_than(group_points, pt1, 4);`	Искомые точки: M N

Получение точек, находящихся выше / ниже / правее / левее заданной оси координат

`points_axes(group_points);`	B правее оси Oy ниже оси Ox C левее оси Oy выше оси Ox D на оси Oy выше оси Ox K левее оси Oy ниже оси Ox T левее оси Oy ниже оси Ox M правее оси Oy выше оси Ox N правее оси Oy выше оси Ox

Получение точек, входящих внутрь заданной прямоугольной зоны

`find_points_in_zone(group_points, 1, -1, 5, 9);`	В заданной области лежат точки: M N

Лабораторная работа №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.

```
class Point {

    constructor(name, x, y) {
        this.pnt = {};
        this.pnt["name"] = name;
        this.pnt["x_cord"] = x;
        this.pnt["y_cord"] = y;
    }

    read_info(flag = 1){
        let string = "";
        for (let j in this.pnt) {
            string += j + ": " + this.pnt[j] + " ";
```

```
        }
        if (flag)
            console.log(string);
        return string;
    }
}

class Section{

    constructor(point1, point2) {
        this.pnt1 = point1;
        this.pnt2 = point2;
        this.name = this.pnt1["name"] + this.pnt2["name"];
    }

    read_info(){
        let str = "Отрезок " + this.name + " задан точками: " +
this.pnt1.read_info(0);
        str += this.pnt2.read_info(0);
        console.log(str);
    }

    find_dist(x1, y1, x2, y2){
        let x = x2 - x1;
        let y = y2 - y1;
        let dist = Math.sqrt(x * x + y * y);
        return dist;
    }

    sect_length(){
        let x1 = this.pnt1.pnt["x_cord"];
        let y1 = this.pnt1.pnt["y_cord"];
        let x2 = this.pnt2.pnt["x_cord"];
        let y2 = this.pnt2.pnt["y_cord"];
        let dist = this.find_dist(x1, y1, x2, y2);
        console.log("�"лина отрезка: " + dist);
    }
}
```

Тесты:

Вывод информации о точках и отрезках на экран:

```
let p1 = new Point("A", 1, 1);
let p2 = new Point("B", 3, 4);
let s1 = new Section(p1, p2);

s1.read_info();

let p11 = new Point("C", -2, 0);
let p22 = new Point("D", 13, -4);
let s11 = new Section(p11, p22);

s11.read_info();
```

```
Отрезок AB задан точками: name: A x_cord: 1 y_cord: 1 name: B x_cord: 3 y_cord: 4

Отрезок CD задан точками: name: C x_cord: -2 y_cord: 0 name: D x_cord: 13 y_cord: -4
```

Нахождение длины отрезка:

```
s1.sect_length();
s11.sect_length();
```

```
Длина отрезка AB: 3.60555127546398
Длина отрезка CD: 15.52417469626002
```


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

```
class Triangle{

    constructor(name, len1, len2, len3) {
        this.len1 = len1;
        this.len2 = len2;
        this.len3 = len3;
        this.name = name;
    }

    check_exist(){
        if ((this.len1 + this.len2 > this.len3) && (this.len1 + this.len3 >
this.len2) && (this.len2 + this.len3 > this.len1)){
            console.log("Треугольник существует");
            return 1;
        }
        else{
            console.log("Треугольник НЕ существует");
            return 0;
        }
    }

    perimetr(flag = 1){
        let len = this.len1 + this.len2 + this.len3;
        if(flag){
            console.log("Перимет треугольника = " + len);
        }
        return len;
    }

    square(flag = 1){
        let p = this.perimetr(0) / 2;
        let s = Math.sqrt(p * (p - this.len1) * (p - this.len2) * (p -
this.len3));
        if(flag){
            console.log("Площадь треугольника = " + s);
        }
        return s;
    }
```

```
        is_rect(){
            let a = this.len1;
            let b = this.len2;
            let c = this.len3;
            if ((a * a + b * b == c * c) || (a * a + c * c == b * b) || (c * c + b * b
    == a * a)){
                console.log("Треуголньник прямоугольный!" );
            }
            else{
                console.log("Треуголньник не является прямоугольным!");
            }
        }
    }
```

Тесты:

Метод проверки возможности существования треугольника с такими сторонами:

`let tr1 = new Triangle("ABC", 3, 4, 5)` `tr1.check_exist()`	`Треугольник существует`

Метод получения периметра треугольника

`tr1.perimetr();`	`Периметр треугольника = 12`

Метод получения площади треугольника

`tr1.square();`	`Площадь треугольника = 6`

Метод для проверки факта: является ли треугольник прямоугольным

`tr1.is_rect();`	`Треуголньник прямоугольный!`

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.

```
let sec = 1;

function proc(interv){
    let interval = setInterval(() => {
        console.log(sec);
        if(sec == 10){
            interv = 1000;
            clearInterval(interval); // остановка работы setInterval
            proc(interv);
        }
        else if (sec == 20){
            interv = 2000;
            sec = 0;
            clearInterval(interval); // остановка работы setInterval
            proc(interv);
        }
        sec++;
    }, interv);
}

proc(2000);
```

Тесты:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1 2 3 4…

Вывод: Во время выполнения лабораторной работы, я изучила особенности синтаксиса языка Java Script, работы со строками, объектами, массивами и функциями на данном языке. Научилась создавать и запускать проекты через Visual Studio Code.