



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 курсу: «Архитектура ЭВМ»

Студент:

Ле Ни Куанг

Группа:

ИУ7и-56Б

Преподаватель:

Попов А. Ю.

Москва

2020

## Цель работы

Ознакомиться с базовым синтаксисом javascript, объектно-ориентированным программированием на javascript.

# Задача 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

```
1 class Child {
2     constructor(name, age) {
3         this.name = name;
4         this.age = age;
5     }
6
7     update(age) {
8         this.age = age;
9     }
10
11     str() {
12         return this.name + " | " + this.age;
13     }
14 }
```

```

15
16 class Store {
17     constructor() {
18         this.data = {};
19     }
20
21     create(name, age) {
22         if (!this.data[name])
23             this.data[name] = new Child(name, age);
24     }
25
26     printLine(key) {
27         this.i++;
28         let line = this.i + " | " + this.data[key].str();
29         console.log(line);
30     }
31
32     read() {
33         this.i = 0;
34         for (let key in this.data) {
35             this.printLine(key);
36         }
37     }
38
39     update(name, age) {
40         this.data[name].age = age;
41     }
42
43     del(name, age) {
44         delete this.data[name];
45     }
46
47     getOldest() {
48         this.i = 0;
49         let ma = 0;
50         for (let key in this.data) {
51             if (this.data[key].age > ma)
52                 ma = this.data[key].age;
53         }
54
55         for (let key in this.data) {
56             if (this.data[key].age == ma)
57                 this.printLine(key);
58         }
59     }
60
61     filter(filterObj) {
62         this.i = 0;

```


```

63         for (let key in this.data) {
64             if (filterObj.isPass(this.data[key]))
65                 this.printLine(key);
66         }
67     }
68
69     findMaxLength() {
70         this.i = 0;
71         let ma = 0;
72         for (let key in this.data) {
73             if (this.data[key].name.length > ma)
74                 ma = this.data[key].name.length;
75         }
76
77         for (let key in this.data) {
78             if (this.data[key].name.length == ma)
79                 this.printLine(key);
80         }
81     }
82 }
83
84 class FilterAge {
85     constructor(mi, ma) {
86         this.mi = mi;
87         this.ma = ma;
88     }
89
90     isPass(child) {
91         return child.age >= this.mi && child.age <= this.ma;
92     }
93 }
94
95 class FilterFirstLetter {
96     constructor(letters) {
97         this.letters = letters;
98     }
99
100     isPass(child) {
101         for (let i in this.letters) {
102             if (child.name[0] === this.letters[i])
103                 return true;
104         }
105         return false;
106     }
107 }

```

# Tect 1

```
1 let Data = new Store;
2 Data.create("A", 10);
3 Data.create("A", 3);
4 Data.create("B", 7);
5 Data.create("C", 3);
6 Data.create("D", 4);
7 Data.create("E", 8);
8 Data.create("Cdfd", 6);
9
10 Data.update("A", 4);
11 Data.del("B");
12
13 Data.read();
14
15 console.log("\nFilter_Age");
16 Data.filter(new FilterAge(4, 9));
17
18 console.log("\nFilter_first_letter_'C'");
19 Data.filter(new FilterFirstLetter("C"));
20
21 console.log("\nFilter_first_letter_is_vowel");
22 Data.filter(new FilterFirstLetter("AaEeUuOoIi"));
23
24 console.log("\nMax_length");
25 Data.create("Dgcg", 6);
26 Data.findMaxLength();
```

5/EVM/1 via  v14.11.0

→ node 1.js

```
1 | A | 4
2 | C | 3
3 | D | 4
4 | E | 8
5 | Cdfd | 6
```

Filter Age

```
1 | A | 4
2 | D | 4
3 | E | 8
4 | Cdfd | 6
```

Filter first letter 'C'

```
1 | C | 3
2 | Cdfd | 6
```

Filter first letter is vowel

```
1 | A | 4
2 | E | 8
```

Max length

```
1 | Cdfd | 6
2 | Dgcg | 6
```

## 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

```
1 class Mark {
2   constructor(mark) {
3     this.data = mark;
4   }
5
6   add(...marks) {
7     this.data.push(marks);
8   }
9
10  average() {
11    return this.data.reduce((a, b) => (a + b)) / this.data.length;
12  }
13
14  str() {
15    return this.data.reduce((a, b) => (a + " " + b), "");
16  }
17
18  len() {
19    return this.data.length;
20  }
21 }
22
23 class Student {
24   constructor(id, group, mark) {
```

```

25         this.id = id;
26         this.group = group;
27         this.mark = new Mark(mark);
28     }
29
30     addMark(...marks) {
31         this.mark.add(marks);
32     }
33
34     getAverage() {
35         return this.mark.average();
36     }
37
38     str() {
39         return this.id + " | " + this.group + " | " + this.mark.str();
40     }
41
42     fullStr() {
43         return this.str() + " | " + this.getAverage();
44     }
45 }
46
47 class Store {
48     constructor() {
49         this.data = {};
50     }
51
52     create(id, group, mark = []) {
53         if (!this.data[id])
54             this.data[id] = new Student(id, group, mark);
55     }
56
57     printLine(id) {
58         console.log(this.data[id].str());
59     }
60
61     read() {
62         for (let id in this.data) {
63             this.printLine(id);
64         }
65     }
66
67     updateGroup(id, group) {
68         this.data[id].group = group;
69     }
70
71     updateMark(id, mark) {
72         this.data[id].mark = mark;

```




```

73     }
74
75     addMark(id, ...marks) {
76         this.data[id].addMark(marks);
77     }
78
79     del(id) {
80         delete this.data[id];
81     }
82
83     getMaxMark() {
84         let ma = 0;
85         for (let id in this.data) {
86             if (this.data[id].mark.len() > ma)
87                 ma = this.data[id].mark.len();
88         }
89
90         for (let id in this.data) {
91             if (this.data[id].mark.len() == ma)
92                 this.printLine(id);
93         }
94     }
95
96     getZeroMark() {
97         for (let id in this.data) {
98             if (this.data[id].mark.len() === 0)
99                 this.printLine(id);
100         }
101     }
102
103     getGroup(group) {
104         let s = new Store;
105         for (let id in this.data) {
106             if (this.data[id].group === group)
107                 s.create(id, group, this.data[id].mark.data)
108         }
109         return s;
110     }
111
112     getStudent(id) {
113         return this.data[id];
114     }
115 }

```

## Tect 2

```
1 let Data = new Store;
2
3 Data.create("ln18iu", "A", [3,4,5,6]);
4 Data.create("ln14iu", "A", [3]);
5 Data.create("pq16iu", "B", [3,4]);
6 Data.create("gh13iu", "C", [3,5,7,6]);
7 Data.create("gh15iu", "C", [3,5,4,6]);
8 Data.create("hg11iu", "D", []);
9 Data.create("hg12iu", "D", [3,4,5,6,7]);
10
11 Data.create("ln18iu", "B", [3,4,5,6]);
12
13 Data.updateGroup("ln14iu", "B");
14 Data.del("gh15iu");
15 Data.read();
16
17 console.log("\nIn_group_D");
18 Data.getGroup("D").read();
19
20 console.log("\nAverage");
21 console.log(Data.getStudent("ln18iu").fullStr());
22
23 console.log("\nMax_marks_in_B");
24 Data.getGroup("B").getMaxMark();
25
26 console.log("\nNo_mark");
27 Data.getZeroMark();
```

5/EVM/1 via  v14.11.0

→ node 2.js

```
ln18iu | A | 3 4 5 6
ln14iu | B | 3
pq16iu | B | 3 4
gh13iu | C | 3 5 7 6
hg11iu | D |
hg12iu | D | 3 4 5 6 7
```

In group D

```
hg11iu | D |
hg12iu | D | 3 4 5 6 7
```

Average

```
ln18iu | A | 3 4 5 6 | 4.5
```

Max marks in B

```
pq16iu | B | 3 4
```

No mark

```
hg11iu | D |
```

## 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

```
1 class Point {
2     constructor(x, y) {
3         this.x = x;
4         this.y = y;
5     }
6
7     str() {
8         return "(" + this.x + "," + this.y + ")";
9     }
10
11     distance(p) {
12         return Math.sqrt((p.x - this.x) ** 2 + (p.y - this.y) ** 2);
13     }
14 }
15
16 class FilterRange {
17     constructor(center, r) {
18         this.center = center;
19         this.r = r;
20     }
21
22     isPass(p) {
23         return this.center.distance(p) <= this.r;
24     }
25 }
```

```

25 }
26
27 class FilterInRectangle {
28     constructor(a, b) {
29         this.mi = new Point(Math.min(a.x, b.x), Math.min(a.y, b.y));
30         this.ma = new Point(Math.max(a.x, b.x), Math.max(a.y, b.y));
31     }
32
33     isPass(p) {
34         return p.x >= this.mi.x && p.y >= this.mi.y &&
35             p.x <= this.ma.x && p.y <= this.ma.y;
36     }
37 }
38
39 class FilterFunc {
40     constructor(point, func) {
41         this.point = point;
42         this.func = func;
43     }
44
45     isPass(p) {
46         return this.func(this.point, p);
47     }
48 }
49
50 class Group {
51     constructor() {
52         this.data = {};
53     }
54
55     create(name, x, y) {
56         if (!this.data[name])
57             this.data[name] = new Point(x, y);
58     }
59
60     read() {
61         for (let name in this.data)
62             this.printPoint(name);
63     }
64
65     update(name, x, y) {
66         this.data[name].x = x;
67         this.data[name].y = y;
68     }
69
70     del(name) {
71         delete this.data[name];
72     }

```

```

73
74     printPoint(name) {
75         console.log(name, this.data[name].str());
76     }
77
78     filter(filterObj) {
79         let g = new Group;
80         for (let name in this.data) {
81             if (filterObj.isPass(this.data[name])) {
82                 g.data[name] = this.data[name];
83             }
84         }
85         return g;
86     }
87
88     printUpDownRightLeft(o) {
89         console.log("Up");
90         this.filter(new FilterFunc(o, (o,p) => p.y > o.y)).read();
91         console.log("Down");
92         this.filter(new FilterFunc(o, (o,p) => p.y < o.y)).read();
93         console.log("Left");
94         this.filter(new FilterFunc(o, (o,p) => p.x < o.x)).read();
95         console.log("Right");
96         this.filter(new FilterFunc(o, (o,p) => p.x > o.x)).read();
97     }
98
99     findMaxDistance() {
100         let key = Object.keys(this.data);
101         let ma = 0; let maxObj = [];
102         let d;
103
104         for (let i = 0; i < key.length - 1; i++) {
105             for (let j = i + 1; j < key.length; j++) {
106                 d = this.data[key[i]].distance(this.data[key[j]]);
107                 if (d > ma) {
108                     ma = d;
109                     maxObj = [[i,j]];
110                 } else if (d == ma) {
111                     maxObj.push([i,j]);
112                 }
113             }
114         }
115
116         console.log(ma);
117         for (let [i,j] of maxObj)
118             console.log(this.data[key[i]].str(), this.data[key[j]].str());
119     }
120 }

```

## Тест 3

```
1 Data.create("D", 5, 4);
2 Data.create("E", 2, 1);
3 Data.create("F", 2, -1);
4
5 Data.read();
6
7 console.log("\n---Max Distance");
8 Data.findMaxDistance();
9 console.log("\n---In circle r");
10 Data.filter(new FilterRange(new Point(3, 1), 3)).read();
11 console.log("\n---Up Down Left Right");
12 Data.printUpDownRightLeft(new Point(3, 4));
13 console.log("\n---In rectangle");
14 Data.filter(new FilterInRectangle(new Point(3, 1), new Point(1, 4))).read();
```

5/EVM/1 via v14.11.0

→ node 3.js

A (3,4)

B (5,7)

C (2,4)

D (5,4)

E (2,1)

F (2,-1)

--- Max Distance

8.54400374531753

(5,7) (2,-1)

--- In circle r

A (3,4)

E (2,1)

F (2,-1)

--- Up Down Left Right

Up

B (5,7)

Down

E (2,1)

F (2,-1)

Left

C (2,4)

E (2,1)

F (2,-1)

Right

B (5,7)

D (5,4)

--- In rectangle

A (3,4)

C (2,4)

E (2,1)

# Задача 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.

```
1 class Point {
2     constructor(x, y) {
3         this.x = x;
4         this.y = y;
5     }
6
7     str() {
8         return "(" + this.x + "," + this.y + ")";
9     }
10
11    log() {
12        console.log(this.str());
13    }
14 }
15
16 class Line {
17     constructor(x1, y1, x2, y2) {
18         this.a = new Point(x1, y1);
19         this.b = new Point(x2, y2);
20     }
21
22    len() {
23        return Math.sqrt((this.a.x - this.b.x) ** 2 + (this.a.y - this.b.y)
24                        ** 2);
25    }
26
27    log() {
28        console.log(this.a.str(), "--", this.b.str(), "└─len:", this.len());
29    }
30 }
```

## Тест 1

```
5/EVM/2 via v14.11.0
→ node 1.js
(1,2)
(1,3) -- (5,6) len: 5
```

## 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

```
1 class Triangle {
2     constructor(a, b, c) {
3         this.a = a;
4         this.b = b;
5         this.c = c;
6     }
7
8     check() {
9         return this.a + this.b > this.c
10            && this.a + this.c > this.b
11            && this.b + this.c > this.a;
12     }
13
14     perimeter() {
15         return this.a + this.b + this.c;
16     }
17 }
```



```

18     area() {
19         let p = this.perimeter() / 2;
20         p = p * (p - this.a) * (p - this.b) * (p - this.c);
21         return Math.sqrt(p);
22     }
23
24     isRight() {
25         let arr = [this.a * this.a, this.b * this.b, this.c * this.c];
26         arr.sort((a, b) => a - b);
27         return arr[0] + arr[1] == arr[2];
28     }
29
30     log() {
31         let c = this.check()
32         console.log("\nCheck_sides", c);
33         if (c) {
34             console.log("Perimeter:", this.perimeter());
35             console.log("Area:", this.area());
36             console.log("Is_right_triangle:", this.isRight());
37         }
38     }
39 }

```

## Tect 2

```

1 let t1 = new Triangle(3, 4, 5);
2 t1.log();
3
4 let t2 = new Triangle(3, 4, 6);
5 t2.log();
6
7 let t3 = new Triangle(3, 4, 8);
8 t3.log();

```

5/EVM/2 via  v14.11.0  
→ node 2.js

```

Check sides true
Perimeter: 12
Area: 6
Is right triangle: true

```

```

Check sides true
Perimeter: 13
Area: 5.332682251925386
Is right triangle: false

```

```

Check sides false

```

## 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.

```
1 // setInterval(() => {
2 //     setTimeout(() => {
3 //         for (let i = 1; i <= 10; i++) console.log(i);
4 //         setTimeout(() => {
5 //             for (let i = 11; i <= 20; i++) console.log(i);
6 //             }, 1000);
7 //     }, 2000);
8 // }, 3000);
9
10
11 function f() {
12     for (let i = 1; i <= 10; i++) {
13         setTimeout(() => {
14             console.log(i);
15         }, i * 200);
16     }
17
18     for (let i = 1; i <= 10; i++) {
19         setTimeout(() => {
20             console.log(i + 10);
21         }, 2000 + i * 100);
22     }
23 }
```

## Тест 3

```
1 f();
2 setInterval(f, 3000);
```

```
5/EVM/2 via v14.11.0
→ node 3.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
```

## Вывод

При выполнении лабораторной работы я изучил основы языка javascript, укрепив свои знания в объектно-ориентированного программирования.