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»

Отчёт
по лабораторной работе № 1

Название: Основные элементы синтаксиса языка JavaScript

Дисциплина: Архитектура ЭВМ

Студент ИУ7-55Б (Группа) _____ (Подпись, дата) Д.О. Склифасовский (И.О. Фамилия)

Преподователь _____ (Подпись, дата) А.Ю. Попов (И.О. Фамилия)

Москва, 2020

Оглавление

Введение

Цель работы: познакомиться с языком программирования JavaScript. Изучить основы ООП.
В ходе лабораторной работы предстоит:

- Выполнить 2 таска, в каждом из которых по 3 задания.

1 Task 1

1.1 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

Решение задания 1

Листинг 1.1: Файл index.js

```
1    "use strict";
2
3    function checkData(Kids, newObj){
```

```
    for (let i = 0; i < Kids.length; i++){
      if (Kids[i] === newObj){
        return 0;
      }
    }
    return 1;
  }

  function CREATE (Kids, newKid){
    if (!checkData(Kids, newKid)) {
      console.log("This child is already in the store.");
    }
    else {
      Kids.push(newKid);
    }
  }

  function READ (Kids, name){
    let checkRes = 0;
    for (let i = 0; i < Kids.length; i++){
      if (Kids[i].surname === name){
        console.log("Age", name, ":", Kids[i].age);
        checkRes = 1;
        break;
      }
    }
    if (!checkRes){
      console.log("This child was not found.");
    }
  }

  function UPDATE(Kids, name, age){
    let checkRes = 0;
    for (let i = 0; i < Kids.length; i++){
      if (Kids[i].surname === name){
        Kids[i].age = age;
        checkRes = 1;
        break;
      }
    }
    if (!checkRes){
      console.log("This child was not found.");
    }
  }

  function DELETE(Kids, name){
```

```
50        let checkRes = 0;
51        let index = -1;
52        for (let i = 0; i < Kids.length; i++){
53          if (Kids[i].surname == name){
54            checkRes = 1;
55            index = i;
56            break;
57          }
58        }
59        if (checkRes){
60          Kids.splice(index, 1);
61        }
62        else{
63          console.log("This child was not found.");
64        }
65      }
66
67      function getAverageAge(Kids){
68        let sum = 0;
69        for (let i = 0; i < Kids.length; i++){
70          sum += Kids[i].age;
71        }
72        return sum / Kids.length;
73      }
74
75      function getInfoOnOldestKid(Kids){
76        let maxAge = 0,
77        index = 0;
78        for (let i = 0; i < Kids.length; i++){
79          if (maxAge < Kids[i].age){
80            maxAge = Kids[i].age;
81            index = i;
82          }
83        }
84        console.log("Oldest child", Kids[index].surname, ", age :", Kids[index].age);
85      }
86
87      function getInfoOnSegment(Kids, indexFrom, indexTo){
88        for (let i = indexFrom; i <= indexTo; i++){
89          console.log("Name:", Kids[i].surname, ", age:", Kids[i].age);
90        }
91      }
92
93      function getInfoByFirstLetter(Kids, letter){
94        let checkRes = 0;
95        for (let i = 0; i < Kids.length; i++){
```

```
        if (Kids[i].surname[0] === letter){
          checkRes = 1;
          console.log("Surname:", Kids[i].surname, ", age:", Kids[i].age);
        }
      }
      if (!checkRes){
        console.log("No one child was found.");
      }
    }

    function getInfoOnCountOfLetters(Kids, count){
      let checkRes = 0;
      for (let i = 0; i < Kids.length; i++){
        if (Kids[i].surname.length > count){
          checkRes = 1;
          console.log("Surname:", Kids[i].surname, ", Age:", Kids[i].age);
        }
      }
      if (!checkRes){
        console.log("No one child was found.");
      }
    }

    function checkInLine(first, second){
      for (let i = 0; i < second.length; i++){
        if (first === second[i]){
          return 1;
        }
      }
      return 0;
    }

    function getInfoByVowel(Kids){
      let vowels = "AEIOU"
      let checkRes = 0;
      for (let i = 0; i < Kids.length; i++){
        if (checkInLine(Kids[i].surname[0], vowels)){
          checkRes = 1;
          console.log("Surname:", Kids[i].surname, ", Age:", Kids[i].age);
        }
      }
      if (!checkRes){
        console.log(""No one child was found."");
      }
    }

```

```
    let Kids = [];
    let frstObject = {"surname" : "Naydenishev", "age" : 20};
    let scndObject = {"surname" : "Syslikov", "age" : 19};
    let thrdObject = {"surname" : "Sklifasovsky", "age" : 20};
    let frthObject = {"surname" : "Orlov", "age" : 15};
    CREATE(Kids, frstObject);
    CREATE(Kids, scndObject);
    CREATE(Kids, thrdObject);
    CREATE(Kids, frthObject);
    READ(Kids, "Orlov");
    //READ(Kids, "Daniil");
    UPDATE(Kids, "Orlov", 16);
    READ(Kids, "Orlov");
    DELETE(Kids, "Orlov");
    //console.log(Kids);
    getInfoOnOldestKid(Kids);
    getInfoOnSegment(Kids, 1, 2);
    getInfoByFirstLetter(Kids, "S");
    getInfoOnCountOfLetters(Kids, 8);
    getInfoByVowel(Kids);
```

1.2 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

Решение задания 2

Листинг 1.2: Файл index.js

```
"use strict";
const StudentsHolder = require("./StudentsHolder");
let Students = new StudentsHolder();
let frstStud = {"group" : 55, "number" : 18491, "marks" : [4, 4, 5]};
let scndStud = {"group" : 55, "number" : 18480, "marks" : [2, 3, 5]};
let thrdStud = {"group" : 53, "number" : 18301, "marks" : [2, 3, 3, 4]};
let frthStud = {"group" : 52, "number" : 18405, "marks" : [4, 4, 4]};
let fifthStud = {"group" : 52, "number" : 18333, "marks" : [5, 5, 5]};
let sixthStud = {"group" : 52, "number" : 18133, "marks" : []};
Students.CREATE(frstStud);
Students.CREATE(scndStud);
Students.CREATE(thrdStud);
Students.CREATE(frthStud);
Students.CREATE(fifthStud);
Students.CREATE(sixthStud);
Students.READ(18491);
Students.READ(18480);
Students.READ(18301);
Students.READ(18405);
Students.READ(18333);
console.log("\n");
```

```
Students.UPDATE(18491, 56, [5, 2, 4]);
Students.READ(18491);
console.log("\n");
Students.DELETE(18480);
console.log(Students.Students);
console.log("\n");
Students.getAverageMarks(18491);
console.log("\n");
Students.getInfoByGroup(52);
console.log("\n");
Students.getInfoByMaxMarks();
console.log("\n");
Students.getInfoByNoMarks();
```

Листинг 1.3: Файл StudentsHolder.js

```
module.exports = class StudentsHolder {
  constructor () {
    this.Students = [];
  }
  getNeedStudent(needNumber) {
    let index = -1;
    for (let i = 0; i < this.Students.length; i++) {
      if (this.Students[i].number === needNumber) {
        index = i;
        break;
      }
    }
    return index;
  }

  checkData(newObj) {
    for (let i = 0; i < this.Students.length; i++) {
      if (this.Students[i].number === newObj.number) {
        return 0;
      }
    }
    return 1;
  }

  CREATE(newStudent) {
    if (!this.checkData(newStudent)) {
      console.log("This student is already present in the repository.");
    }
    else {
      this.Students.push(newStudent);
```

```
31          }
32        }
33
34        READ(needStudent) {
35          let index = this.getNeedStudent(needStudent);
36          if (index === -1) {
37            console.log("This student was not found.");
38          }
39          else {
40            console.log("Group:", this.Students[index].group, "Number:", this.
                Students[index].number, "Marks:", this.Students[index].marks);
41          }
42        }
43
44        UPDATE(needNumber, newGroup, newMarks) {
45          let index = this.getNeedStudent(needNumber);
46          if (index === -1) {
47            console.log("This student was not found.");
48          }
49          else {
50            this.Students[index].group = newGroup;
51            this.Students[index].marks = newMarks;
52          }
53        }
54
55        DELETE(needNumber) {
56          let index = this.getNeedStudent(needNumber);
57          if (index === -1) {
58            console.log("This student was not found.");
59          }
60          else {
61            this.Students.splice(index, 1);
62          }
63        }
64
65        sum(arr) {
66          let res = 0;
67          for (let i = 0; i < arr.length; i++) {
68            res += arr[i];
69          }
70          return res;
71        }
72
73        getAverageMarks(needNumber) {
74          let index = this.getNeedStudent(needNumber);
75          if (index === -1) {
```

```
76            console.log("This student was not found.");
77          }
78          else {
79            let sum = this.sum(this.Students[index].marks);
80            if (this.Students[index].marks.length != 0) {
81              console.log(sum / this.Students[index].marks.length);
82            }
83            else {
84              console.log("This student has no grades.");
85            }
86          }
87        }
88
89        getInfoByGroup(needGroup) {
90          for (let i = 0; i < this.Students.length; i++) {
91            if (this.Students[i].group === needGroup) {
92              this.READ(this.Students[i].number);
93            }
94          }
95        }
96
97        getInfoByMaxMarks() {
98          let index = -1;
99          let maxMarks = 0;
100         for (let i = 0; i < this.Students.length; i++) {
101           if (maxMarks < this.Students[i].marks.length) {
102             maxMarks = this.Students[i].marks.length;
103             index = i;
104           }
105         }
106         if (index != -1) {
107           this.READ(this.Students[index].number);
108         }
109         else {
110           console.log("The required student was not found");
111         }
112       }
113
114       getInfoByNoMarks() {
115         for (let i = 0; i < this.Students.length; i++) {
116           if (this.Students[i].marks.length === 0) {
117             this.READ(this.Students[i].number);
118           }
119         }
120       }
121     }
```

1.3 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

Решение задания 3

Листинг 1.4: Файл index.js

```
"use strict";
const Point = require("./Point");
const PointHandler = require("./PointHandler");
const Rectangle = require("./Rectangle");
let frstPoint = new Point("A", 5, 5);
let scndPoint = new Point("B", 2, 2);
let thrdPoint = new Point("C", -1, 1);
let frthPoint = new Point("D", 1, -1);
let fifthPoint = new Point("E", 5, -5);
let sixthPoint = new Point("F", 10, 10);
let Points = new PointHandler();
Points.CREATE(frstPoint);
Points.CREATE(scndPoint);
Points.CREATE(thrdPoint);
Points.CREATE(frthPoint);
Points.CREATE(fifthPoint);
Points.CREATE(sixthPoint);
Points.READ("A");
Points.READ("B");
```

```
20    Points.READ("C");
21    Points.READ("D");
22    Points.READ("E");
23    Points.READ("F");
24    console.log("\n");
25    Points.UPDATE("B", -2, -2);
26    Points.READ("B");
27    console.log("\n");
28    Points.DELETE("F");
29    console.log(Points.Points);
30    console.log("\n");
31    Points.findMaxDistance();
32    console.log("\n");
33    Points.findPointsByNewPoint(0, 0, 4);
34    console.log("\n");
35    Points.findPointsAboutGivenAxis(0);
36    Points.findPointsAboutGivenAxis(1);
37    console.log("\n");
38    let firstVertex = new Point("", -4, 4);
39    let secondVertex = new Point("", 4, 4);
40    let thirdVertex = new Point("", 4, -4);
41    let fourthVertex = new Point("", -4, -4);
42    let curRect = new Rectangle(firstVertex, secondVertex, thirdVertex,
          fourthVertex);
43    Points.findPointsInrectangle(curRect);
```

Листинг 1.5: Файл Point.js

```
1     module.exports = class Point {
2       constructor(name, x, y) {
3         this.name = name;
4         this.x = x;
5         this.y = y;
6       }
7     }
```

Листинг 1.6: Файл PointHandler.js

```
1     const Point = require("./Point");
2     module.exports = class PointHandler {
3       constructor () {
4         this.Points = [];
5         this.length = this.Points.length;
6       }
7
8       checkData(curPoint) {
9         for (let i = 0; i < this.length; i++) {
```

```
10          if (curPoint.name === this.Points[i].name) {
11            return 0;
12          }
13        }
14        return 1;
15      }
16      getPointByName(name) {
17        let index = -1;
18        for (let i = 0; i < this.length; i++) {
19          if (this.Points[i].name === name) {
20            index = i;
21            break;
22          }
23        }
24        return index;
25      }
26      CREATE(curPoint) {
27        if (this.checkData(curPoint)) {
28          this.Points.push(curPoint);
29          this.length++;
30        }
31        else {
32          console.log("This point already exists.");
33        }
34      }
35      READ(name) {
36        let index = this.getPointByName(name);
37        if (index === -1) {
38          console.log("This point already exists.");
39        }
40        else {
41          console.log("Point: " + this.Points[index].name + ", x: " + this.Points[
              index].x + ", y: " + this.Points[index].y + ".");
42        }
43      }
44      UPDATE(name, x, y) {
45        let index = this.getPointByName(name);
46        if (index === -1) {
47          console.log("This point already exists.");
48        }
49        else {
50          this.Points[index].x = x;
51          this.Points[index].y = y;
52        }
53      }
54      DELETE(name) {
```

```
55          let index = this.getPointByName(name);
56          if (index === -1) {
57            console.log("This point already exists.");
58          }
59          else {
60            this.Points.splice(index, 1);
61            this.length--;
62          }
63        }
64        getDistance(frstPoint, scndPoint) {
65          return Math.sqrt(Math.pow(scndPoint.x - frstPoint.x, 2) + Math.pow(
              scndPoint.y - frstPoint.y, 2))
66        }
67        findMaxDistance() {
68          let frstPoint = this.Points[0],
69          scndPoint = this.Points[1];
70          let maxDistance = this.getDistance(frstPoint, scndPoint);
71          for (let i = 0; i < this.length - 1; i++) {
72            for (let j = i + 1; j < this.length; j++) {
73              let curDistance = this.getDistance(this.Points[i], this.Points[j])
74              if (curDistance > maxDistance) {
75                maxDistance = curDistance;
76                frstPoint = this.Points[i];
77                scndPoint = this.Points[j];
78              }
79            }
80          }
81          console.log("Maximum distance = " + maxDistance + " between points:")
82          this.READ(frstPoint.name);
83          this.READ(scndPoint.name);
84        }
85        findPointsByNewPoint(x, y, constant) {
86          for (let i = 0; i < this.length; i++) {
87            if (this.getDistance(new Point("CheckDistance", x, y), this.Points[i]) <=
                  constant) {
88              this.READ(this.Points[i].name);
89            }
90          }
91        }
92        // 0 - x, 1 - y
93        findAboutX() {
94          console.log("Above Y:");
95          for (let i = 0; i < this.length; i++) {
96            if (this.Points[i].y > 0) {
97              this.READ(this.Points[i].name);
98            }
```

```
        }
        console.log("Below Y:");
        for (let i = 0; i < this.length; i++) {
          if (this.Points[i].y < 0) {
            this.READ(this.Points[i].name);
          }
        }
      }
      findAboutY() {
        console.log("To the right X:");
        for (let i = 0; i < this.length; i++) {
          if (this.Points[i].x > 0) {
            this.READ(this.Points[i].name);
          }
        }
        console.log("To the left X:");
        for (let i = 0; i < this.length; i++) {
          if (this.Points[i].x < 0) {
            this.READ(this.Points[i].name);
          }
        }
      }
      findPointsAboutGivenAxis(axis) {
        if (!axis) {
          this.findAboutX();
        }
        else {
          this.findAboutY();
        }
      }
      findMaxCoords(curRect) {
        let maxX = curRect.curRectangle[0].x,
        maxY = curRect.curRectangle[0].y;
        for (let i = 0; i < 4; i++) {
          if (maxX < curRect.curRectangle[i].x) {
            maxX = curRect.curRectangle[i].x;
          }
          if (maxY < curRect.curRectangle[i].y) {
            maxY = curRect.curRectangle[i].y;
          }
        }
        return new Point("", maxX, maxY);
      }
      findMinCoords(curRect) {
        let minX = curRect.curRectangle[0].x,
        minY = curRect.curRectangle[0].y;
```

```
        for (let i = 0; i < 4; i++) {
          if (minX > curRect.curRectangle[i].x) {
            minX = curRect.curRectangle[i].x;
          }
          if (minY > curRect.curRectangle[i].y) {
            minY = curRect.curRectangle[i].y;
          }
        }
        return new Point("", minX, minY);
      }
      IsInRect(curRect, curPoint) {
        let maxPoint = this.findMaxCoords(curRect);
        let minPoint = this.findMinCoords(curRect);
        if (curPoint.x < maxPoint.x && curPoint.x > minPoint.x &&
        curPoint.y < maxPoint.y && curPoint.y > minPoint.y) {
          return 1;
        }
        return 0;
      }
      findPointsInrectangle(curRect) {
        for (let i = 0; i < this.length; i++) {
          if (this.IsInRect(curRect, this.Points[i])) {
            this.READ(this.Points[i].name);
          }
        }
      }
    }
```

Листинг 1.7: Файл Rectangle.js

```
const Point = require("./Point")
module.exports = class Rectangle {
  constructor(firstPoint, secondPoint, thirdPoint, fourthPoint) {
    this.curRectangle = [];
    this.curRectangle.push(firstPoint);
    this.curRectangle.push(secondPoint);
    this.curRectangle.push(thirdPoint);
    this.curRectangle.push(fourthPoint);
  }
}
```

2 Task 2

2.1 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.
Решение задания 1

Листинг 2.1: Файл index.js

```
1    "use strict";
2    const Point = require("./Point");
3    const Section = require("./Section");
4    let checkSection = new Section();
5    checkSection.init(0, 0, 4, 0);
6    checkSection.printSection();
7    let length = checkSection.findLength();
8    console.log(length);
```

Листинг 2.2: Файл Point.js

```
1    "use strict";
2    module.exports = class Point {
3      constructor() {
4        this.x = 0;
5        this.y = 0;
6      }
7
8      init(x, y) {
```

```
9         this.x = x;
10        this.y = y;
11      }
12
13      printPoint() {
14        console.log("Point: " + this.x + ";" + this.y);
15      }
16    }
```

Листинг 2.3: Файл Section.js

```
1   "use strict";
2   const Point = require("./Point");
3   module.exports = class Section {
4     constructor() {
5       this.firstPoint = new Point();
6       this.secondPoint = new Point();
7     }
8
9     init(x1, y1, x2, y2) {
10      this.firstPoint.init(x1, y1);
11      this.secondPoint.init(x2, y2);
12    }
13
14    printSection() {
15      console.log("Section: ");
16      this.firstPoint.printPoint();
17      this.secondPoint.printPoint();
18    }
19
20    findLength() {
21      return Math.sqrt(Math.pow(this.secondPoint.x - this.firstPoint.x , 2) +
            Math.pow(this.secondPoint.y - this.firstPoint.y , 2));
22    }
23  }
```

2.2 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

Решение задания 2

Листинг 2.4: Файл index.js

```
1   "use strict";
2   const Triangle = require("./Tiangle");
3   let checkTriangle = new Triangle();
4   checkTriangle.init(5, 5, 6);
5   console.log(checkTriangle.isExist());
6   console.log(checkTriangle.getPerimeter());
7   console.log(checkTriangle.getArea());
8   console.log(checkTriangle.isRectangular());
9   console.log("\n");
10  let checkNewTriangle = new Triangle();
11  checkNewTriangle.init(1, 1, 4);
12  console.log(checkNewTriangle.isExist());
13  console.log("\n");
14  checkNewTriangle.init(3, 4, 5);
15  console.log(checkNewTriangle.isExist());
16  console.log(checkNewTriangle.isRectangular());
```

Листинг 2.5: Файл Triangle.js

```
1   "use strict";
2
3   module.exports = class Triangle {
```

```
4       constructor() {
5         this.lenA = 0;
6         this.lenB = 0;
7         this.lenC = 0;
8       }
9
10      init(len1, len2, len3) {
11        this.lenA = len1;
12        this.lenB = len2;
13        this.lenC = len3;
14      }
15
16      isExist() {
17        if (this.lenA + this.lenB > this.lenC &&
18        this.lenA + this.lenC > this.lenB &&
19        this.lenB + this.lenC > this.lenA) {
20          return 1;
21        }
22        return 0;
23      }
24
25      getPerimeter() {
26        if (this.isExist()) {
27          return this.lenA + this.lenB + this.lenC;
28        }
29        else {
30          console.log("There is no triangle.");
31          return -1;
32        }
33      }
34
35      getArea() {
36        if (this.isExist()) {
37          let perimeter = this.getPerimeter();
38          return Math.sqrt(perimeter * (perimeter - this.lenA) * (perimeter - this.
                lenB) * (perimeter - this.lenC));
39        }
40        else {
41          console.log("There is no triangle.");
42          return -1;
43        }
44      }
45
46      checkSide(firstLen, secondLen, thirdLen) {
47        if (Math.pow(firstLen, 2) + Math.pow(secondLen, 2) === Math.pow(thirdLen,
              2)) {
```

```
48            return 1;
49          }
50          return 0;
51        }
52
53        isRectangular() {
54          if (this.isExist()) {
55            if (this.checkSide(this.lenA, this.lenB, this.lenC) || this.checkSide(
                  this.lenA, this.lenC, this.lenB) || this.checkSide(this.lenB, this.
                  lenC, this.lenA)) {
56              return 1;
57            }
58            else {
59              return 0;
60            }
61          }
62          else {
63            console.log("There is no triangle.");
64            return -1;
65          }
66        }
67      }
```

2.3 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.

Решение задания 2

Листинг 2.6: Файл index.js

```
"use strict";

function checkInterval(startNumber, endNumber, intervalTime, i) {
  let interval = setInterval(() => {
    console.log(startNumber);
    startNumber++;
    if (startNumber == endNumber + 1) {
      clearInterval(interval);
      i++;
      let start = i % 2 * 10 + 1;
      let end = start + 9;
      let intTime = 1000;
      if (i % 2 == 0) {
        intTime = 2000;
      }
      checkInterval(start, end, intTime, i);
    }
  }, intervalTime);
}

checkInterval(1, 10, 2000, 0);
```

Заключение

В ходе выполнения данной лабораторной работой я познакомился с языком программирования JavaScript и изучил основы ООП.