



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»

Тема Основы JavaScript

Студент Романов А.В.

Группа ИУ7-53Б

Преподаватели Попов А.Ю.

Отчет по разделу #1

Цель работы

Ознакомиться с особенностями и базовыми принципами программирования на **JavaScript**.

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:

- CRUD 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**

ex_01_01.js

```
"use strict";

const VOWELS = [ 'A', 'E', 'I', 'O', 'U', 'Y' ];

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

class childrenStorage {
  constructor() {
    this.storage = [];
  }

  read(surname) {
    return this.storage.find(child => child.surname === surname);
  }

  create(surname, age) {
    if (this.read(surname)) {
      throw "Child exists."
    }

    this.storage.push(new Child(surname, age));
  }

  update(surname, new_surname, new_age) {
    let child = this.read(surname);
    if (!child) {
      throw "Child doesn't exists."
    }

    if (this.read(new_surname)) {
      throw "Child with this surname exists."
    }

    child.surname = new_surname;
```

```

        child.surname = new_surname;
        child.age = new_age;
    }

    delete(surname) {
        if (!this.read(surname)) {
            throw "Child doesn't exists."
        }

        this.storage = this.storage.filter(child => child.surname !== surname);
    }

    avg_age() {
        if (!this.storage.length) {
            throw "Storage is empty."
        }

        return this.storage.reduce((acc, child) => child.age + acc, 0) / this.storage.length;
    }

    oldest() {
        return this.storage.reduce((acc, child) => child.age > acc.age ? child : acc, this.storage[0])
    }

    at_interval(left, right) {
        return this.storage.filter(child => child.age >= left && child.age <= right);
    }

    fst_symbol(symbol) {
        return this.storage.filter(child => child.surname[0] === symbol);
    }

    surname_longer_than(size) {
        return this.storage.filter(child => child.surname.length > size)
    }

    vowel_fst() {
        return this.storage.filter(child => VOWELS.find(symb => symb === child.surname[0]))
    }
}

function main() {
    let p = new Child("Perestoronin", 20);
    let storage = new childrenStorage();

    storage.create("Perestoronin", 20);
    storage.update("Perestoronin", "Peperonin", 20)
    storage.create("Perestoronin", 15);
    storage.create("Perestoronin111", 25);
    storage.create("XPerestoronin111", 25);

    console.log(storage.avg_age());
    console.log(storage.oldest());
    console.log(storage.at_interval(15, 23));
    console.log(storage.fst_symbol("P"));
    console.log(storage.surname_longer_than(11));
    console.log(storage.vowel_fst());
}

main()

```

Результаты тестирования

Все тесты были пройдены успешно.

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:

- CRUD 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**

ex_01_02.js

```
"use strict";

class Student {
  constructor(id, group, rating_list) {
    this.id = id;
    this.group = group;
    this.rating_list = rating_list;
  }
}

class studentsStorage {
  constructor() {
    this.storage = [];
  }

  read(id) {
    return this.storage.find(student => student.id === id);
  }

  create(id, group, rating_list) {
    if (this.read(id)) {
      throw "Student exists."
    }

    this.storage.push(new Student(id, group, rating_list));
  }

  update(id, new_id, new_group, new_rating_list) {
    let student = this.read(id);
    if (!student) {
      throw "Student doesn't exists."
    }

    if (this.read(new_id)) {
      return false;
    }

    student.id = new_id;
    student.group = new_group;
    student.rating_list = new_rating_list;
  }

  delete(id) {
    if (!this.read(id)) {
      throw "Student doesn't exists."
    }
  }
}
```

```

    this.storage = this.storage.filter(student => student.id !== id);
  }

  avg_rating(id) {
    const student = this.storage.find(student => student.id === id);
    if (!student) {
      throw "This student doesn't exists."
    }

    if (!student.rating_list.length) {
      throw "This student has no rating."
    }

    return student.rating_list.reduce((acc, mark) => acc + mark, 0) / student.rating_list.length;
  }

  group_info(group) {
    return this.storage.filter(student => student.group === group);
  }

  max_marks() {
    if (!this.storage.length) {
      throw "Storage is empty."
    }

    return this.storage.reduce((acc, student) => student.rating_list.length >
      acc.rating_list.length ? student : acc, this.storage[0])
  }

  no_rating() {
    return this.storage.filter(student => !student.rating_list.length);
  }
}

function main() {
  let p = new Student(1, "IU7-33B", [1, 3, 3, 7]);
  let storage = new studentsStorage();

  storage.create(1, "IU7-33B", [1, 4, 8, 8]);
  storage.update(1, 1, "IU7-33B", [2, 3, 4]);
  storage.create(2, "IU7-33B", [2, 2, 8]);
  storage.create(5, "IU7-53B", [2, 2, 8]);
  storage.create(9, "IU7-53B", [6, 6, 6, 6]);
  storage.create(17, "IU7-53B", []);

  console.log(storage);
  console.log(storage.read(1));
  console.log(storage.avg_rating(1));
  console.log(storage.group_info("IU7-33B"));
  console.log(storage.max_marks());
  console.log(storage.no_rating());
}

main();

```

Результаты тестирования

Все тесты были пройдены успешно.

Задание 3

Условие

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:

- CRUD 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**

ex_01_03.ts

```
"use strict";

class Point {
  constructor(name, x, y) {
    this.name = name;
    this.x = x;
    this.y = y;
  }
}

class pointStorage {
  constructor() {
    this.storage = [];
  }

  read(name) {
    return this.storage.find(point => point.name === name);
  }

  create(name, x, y) {
    if (this.read(name)) {
      throw "Point exists."
    }

    this.storage.push(new Point(name, x, y));
  }

  update(name, new_name, x, y) {
    let point = this.read(name);
    if (!point) {
      throw "Point doesn't exists."
    }

    point.name = name;
    point.x = x;
    point.y = y;
  }

  delete(name) {
    if (!this.read(name)) {
      throw "Point doesn't exists."
    }

    this.storage = this.storage.filter(point => point.name !== name);
  }

  get_dist(p1, p2) {
    let dx = p1.x - p2.x;
    let dy = p1.y - p2.y;
    return Math.sqrt(dx * dx + dy * dy);
  }

  max_distance() {
    return this.storage.reduce((acc1, p1, i) => this.storage.slice(i + 1).reduce((acc2, p2) =>
```

```

        this.get_dist(p1, p2) > acc2.dist ?
        { points: { p1, p2 }, dist: this.get_dist(p1, p2) } : acc2, acc1), {
        points: { p1: this.storage[0], p2: this.storage[1] }, dist: this.get_dist(this.storage[0], this.storage[1])
    })
}

at_interval(point, len) {
    return this.storage.filter(pt => this.get_dist(point, pt) <= len);
}

in_zone(is_x, is_more) {
    return this.storage.filter(is_x ? pt => pt.x >= 0 !== is_more : pt => pt.y >= 0 !== is_more);
}

in_rectangle(x_min, y_min, x_max, y_max) {
    return this.storage.filter(pt => pt.x >= x_min && pt.x <= x_max && pt.y >= y_min && pt.y <= y_max);
}
}

function main() {
    let p = new Point("f1", 5, 10);
    let storage = new pointStorage();

    storage.create("center", 0, 0);
    storage.create("p1", 10, 10);
    storage.create("p2", -10, 10);
    storage.create("p3", 10, -10);
    storage.create("p4", -10, -10);
    storage.create("p5", 5, 5);
    storage.create("p6", 7, 5);
    storage.create("p8", 2, 3);
    storage.create("p9", 1000, 3);

    console.log(storage);
    console.log(storage.max_distance());
    console.log(storage.at_interval(p, 23));
    console.log(storage);
    console.log(storage.in_zone(true, true))
    console.log(storage.in_rectangle(-5, -5, 10, 10));
}

main();

```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы были изучены базовые приемы работы с **JavaScript**, освоены особенности работы с данным языком программирования.

Отчет по разделу #2

Цель работы

Ознакомиться с особенностями объектно-ориентированного программирования в JavaScript.

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**

ex_02_01.js

```
"use strict";

class Point {
  constructor(x, y) {
    this.set(x, y);
  }

  set(x, y) {
    this.x = x;
    this.y = y;
  }

  print() {
    console.log(` Point (x: ${this.x}, y: ${this.y})`)
  }
}

class Line {
  constructor(p1, p2) {
    this.set(p1, p2);
  }

  set(p1, p2) {
    this.p1 = p1;
    this.p2 = p2;
  }

  len() {
    const dx = this.p2.x - this.p1.x;
    const dy = this.p2.y - this.p1.y;
    return Math.sqrt(dx * dx + dy * dy);
  }

  print() {
    console.log(` Line: (`)

    process.stdout.write('\t')
    this.p1.print()
    process.stdout.write('\t')
    this.p2.print()

    console.log(')')
  }
}

function main() {
  let p1 = new Point(1, 5)
  p1.print();

  let p2 = new Point(1, 5)

  let s = new Line(p1, p2);
  s.print();
}

main()
```


Результаты тестирования

Все тесты были пройдены успешно.

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**

ex_02_02.js

```
"use strict";

class Triangle {
  constructor(a, b, c) {
    this.set(a, b, c);
  }

  set(a, b, c) {
    this.a = a;
    this.b = b;
    this.c = c;
  }

  is_exists() {
    return this.a < this.b + this.c && this.b < this.a + this.c && this.c < this.b + this.a;
  }

  perimeter() {
    if (!this.is_exists()) {
      throw "Triangle doesn't exists."
    }

    return this.a + this.b + this.c;
  }

  area() {
    if (!this.is_exists()) {
      throw "Triangle doesn't exists."
    }

    const p = this.perimeter() / 2;
    return Math.sqrt(p * (p - this.a) * (p - this.b) * (p - this.c));
  }

  is_rectangular() {
    if (!this.is_exists()) {
      throw "Triangle doesn't exists."
    }

    const sides = [this.a, this.b, this.c].sort();
    return Math.abs(sides[2] * sides[2] - (sides[1] * sides[1] + sides[0] * sides[0])) < 1e-5;
  }
}

function main()
{
  let t = new Triangle(3, 4, 5);
  console.log(t.perimeter());
  console.log(t.is_exists());
  console.log(t.area());
  console.log(t.is_rectangular());
}

main();
```

Результаты тестирования

Все тесты были пройдены успешно.

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**

ex_02_03.ts

```
"use strict";

let exec = function(cnt, delay) {
    console.log(++cnt);

    if (cnt === 10) {
        delay = 2000;
    }

    if (cnt === 20) {
        cnt = 0;
        delay = 1000;
    }

    setTimeout(exec, delay, cnt, delay);
}

function main() {
    exec(0, 1000);
}

main();
```

Результаты тестирования

Все тесты были пройдены успешно.

Вывод

В результате работы были изучены особенности работы с принципами ООП в **JavaScript**, применены на практике приемы и особенности разработки.