ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ

Институт Принтмедиа и информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 6

Дисциплина: Технологии кроссплатформенного программирования

Тема: Интерактивность в JS

Выполнил: студент группы 201-725
Филиппов К. Д.

Дата, подпись _____ _____
(Дата) (Подпись)

Проверил: _____ _____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____
(Дата) (Подпись)

Москва, 2022

Тема работы: Изучение приемов создания интерактивных приложений в JS.
Цель работы: Изучить приёмы создания интерактивных приложений с использованием Canvas в языке JS.

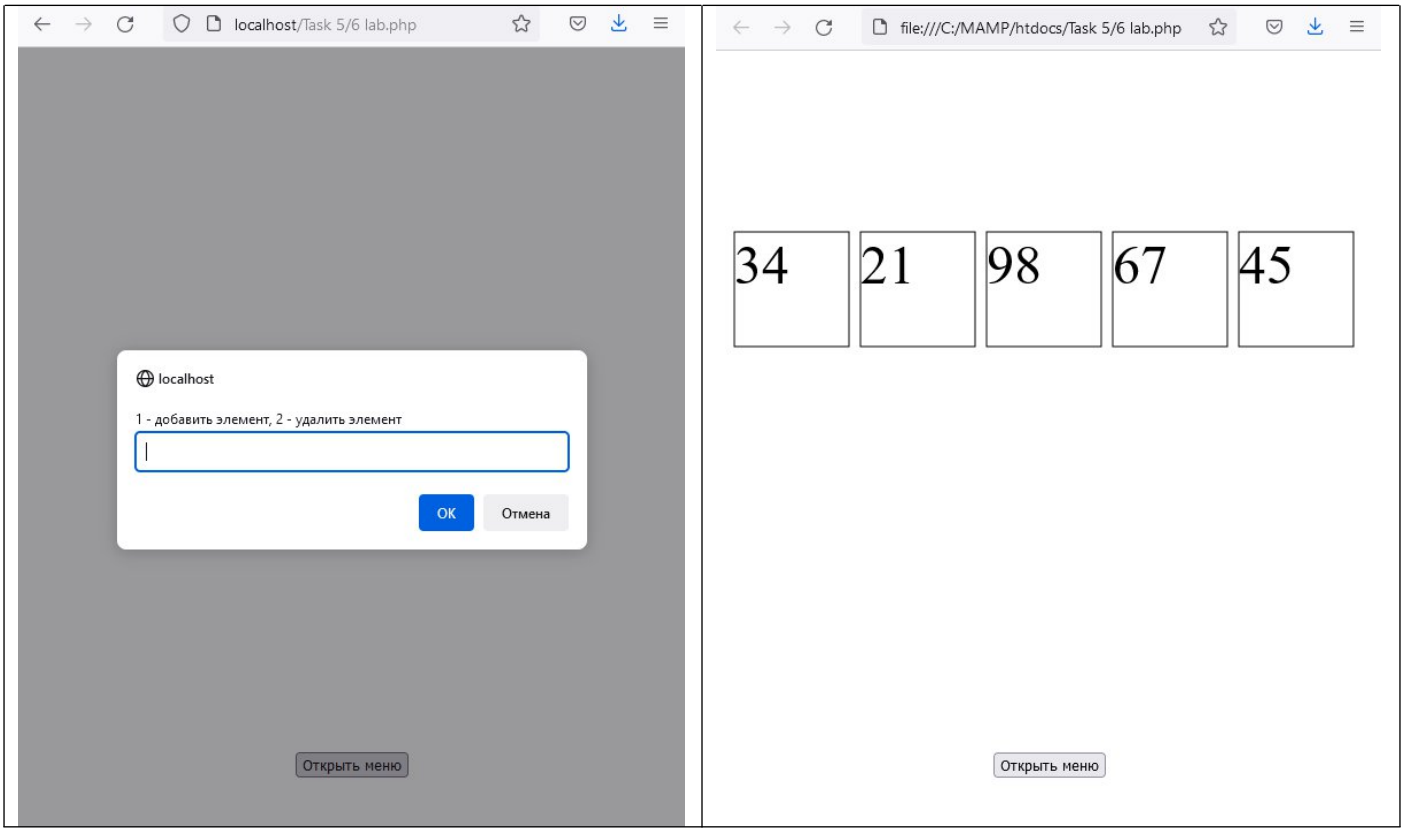
Ход работы

Canvas — это элемент HTML5 для создания растрового двухмерного изображения при помощи скриптов на языке JavaScript.
Задание 1. Для объектно-ориентированной программной модели **стека** разработаем пользовательский графический интерфейс.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>1</title>
    <meta charset="UTF-8">
    <script type="application/javascript">
    /*
    Для объектно-ориентированной программной модели СТЕКА разработать пользовательский
графический интерфейс.
    */
        let stack = {
            arr: [],
            height: 100,
            width: 100,
            space: 10,

            Push: function (x) {
                this.arr.unshift(x);
            },

            Pop: function () {
                if (this.arr.length > 0) {
                    return this.arr.shift();
                } else {
                    alert("Стек пуст!");
                }
            },

            Draw: function () {
                var canvas = document.getElementById('canvas');
                var ctx = canvas.getContext('2d');
                ctx.clearRect(0, 0, canvas.width, canvas.height);
                for (let i = 0; i < this.arr.length; i++) {
                    ctx.strokeRect(this.space * (i + 1) + this.width * i, 150,
this.width, this.height);
                    ctx.fillText(this.arr[i], this.space * (i + 1) + this.width * i,
this.width + 95);
                }
            }
        }

    </script>
</head>
<body>
<div style="text-align: center;">
    <canvas id="canvas" width="800" height="600"></canvas>
</div>
<div style="text-align: center;">
    <input type="button" value="Открыть меню" onclick="menu();"/>
</div>
<script>
```

```
    document.getElementById('canvas').getContext('2d').font = '48px serif';


    let menu = function () {
        const choice = prompt("1 - добавить элемент, 2 - удалить элемент");

        switch (choice) {
            case "1":
                stack.Push(prompt("Введите число"));
                stack.Draw();
                break;
            case "2":
                alert("Был удалён элемент " + stack.Pop());
                stack.Draw();
                break;
        }
    }

</script>
</body>
</html>
```

Результат выполнения программы:

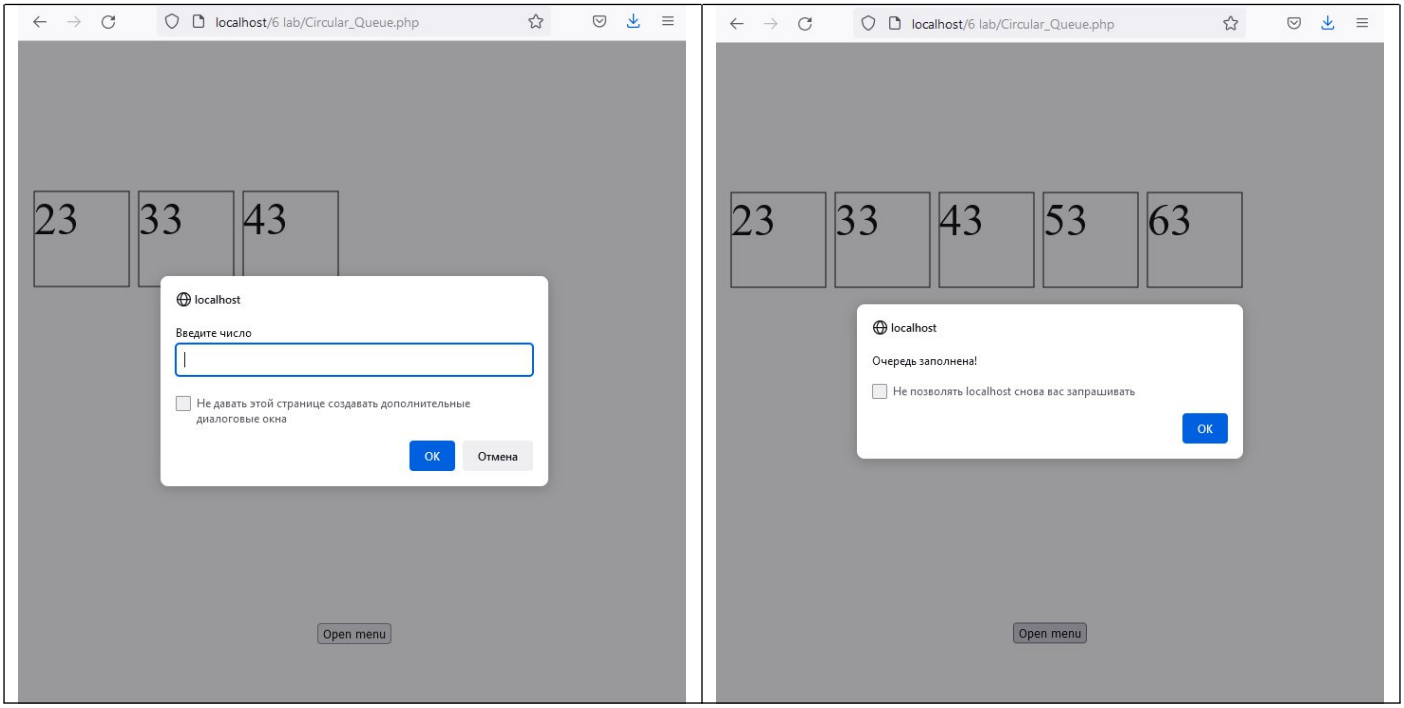


Задание 2. Для объектно-ориентированной программной модели **циклической очереди** разработаем пользовательский графический интерфейс.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>2</title>
    <meta charset="UTF-8">
    <script type="application/javascript">
    /*
    Для объектно-ориентированной программной модели ЦИКЛИЧЕСКОЙ ОЧЕРЕДИ разработать
пользовательский графический интерфейс.
    */

        let Queue = {
            arr: [],
            rear: -1,
            front: -1,
            size: 0,
            height: 100,
            width: 100,
            space: 10,
            realSize: 0,

            SetSize: function (size) {
                this.size = size;
            },

            Push: function (x) {
                if (this.front === 0 && this.rear === this.size - 1 || this.front ===
this.rear + 1) {
                    alert("Очередь заполнена!");
                } else {
                    if (this.front === -1) {
                        this.front = 0;
                    }
                    this.rear = (this.rear + 1) % this.size;
                    this.arr[this.rear] = x;
                    this.realSize++;
                }
            },

            Pop: function () {
                if (this.front === -1) {
                    alert("Очередь пуста!");
                } else {
                    if (this.front === this.rear) {
                        this.front = -1;
                        this.rear = -1;
                    } else {
                        this.front = (this.front + 1) % this.size;
                    }
                    this.realSize--;
                    return this.arr.shift();
                }
            },

            Draw: function () {
                var canvas = document.getElementById('canvas');
                var ctx = canvas.getContext('2d');
                ctx.clearRect(0, 0, canvas.width, canvas.height);
                for (let i = 0; i < this.arr.length; i++) {
                    ctx.strokeRect(this.space * (i + 1) + this.width * i, 150,
this.width, this.height);
                    ctx.fillText(this.arr[i], this.space * (i + 1) + this.width * i,
this.width + 95);
```

```
                }
            }
        }

    </script>
</head>
<body>
<div style="text-align: center;">
    <canvas id="canvas" width="800" height="600"></canvas>
</div>
<div style="text-align: center;">
    <input type="button" value="Открыть меню" onclick="menu();"/>
</div>
<script>
    size = parseInt(prompt("Введите размер очереди"));
    Queue.SetSize(size);

    document.getElementById('canvas').getContext('2d').font = '48px serif';


    let menu = function () {
        const choice = prompt("1 - добавить элемент, 2 - удалить элемент");

        switch (choice) {
            case "1":
                Queue.Push(prompt("Введите число"));
                Queue.Draw();
                break;
            case "2":
                alert("Был удалён элемент " + Queue.Pop());
                Queue.Draw();
                break;
        }
    }

</script>
</body>
</html>
```

Результат выполнения программы:



Задание 3. Для объектно-ориентированной программной модели двусвязного списка разработаем пользовательский графический интерфейс.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
    /*
    Для объектно-ориентированной программной модели ДВУСВЯЗНОГО СПИСКА разработать
пользовательский графический интерфейс.
    */
        class Node {

            constructor(left = null, right = null, data) {
                this.left = left;
                this.right = right;
                this.data = data;
            }

        }

        let DoubleLinkedList = {

            rear: null,
            front: null,
            width: 100,
            height: 100,
            space: 10,
            count: 0,

            AddRear: function (data) {
                if ((this.rear == null)) {
                    if (this.front == null) {
                        this.front = new Node(undefined, undefined, data);
                    } else {
                        this.rear = new Node(this.front, undefined, data);
                        this.front.right = this.rear;
                    }
                } else {
                    this.rear.right = new Node(this.rear, undefined, data);
                    this.rear = this.rear.right;
                }
                this.count++;
            },

            AddFront: function (data) {
                if (this.front == null) {
                    if (this.rear == null) {
                        this.front = new Node(undefined, undefined, data);
                    } else {
                        this.front = new Node(undefined, this.rear, data);
                        this.rear.left = this.front;
                    }
                } else {
                    if (this.rear == null) {
                        this.rear = this.front;
                        this.front = new Node(undefined, this.rear, data);
                        this.rear.left = this.front;
                    } else {
                        this.front.left = new Node(undefined, this.front, data);
                        this.front.left.right = this.front;
                        this.front = this.front.left;
```

```
            }
        }
        this.count++;
    },

    DeleteBack: function () {
        if (this.rear == null) {
            if (this.front == null) {
                alert("Список пуст!");
                return;
            } else {
                delete this.front;
            }
        } else {
            if (this.count === 2) {
                delete this.rear;
            } else {
                this.rear = this.rear.left;
                this.rear.right = null;
            }
        }
        this.count--;
    },

    DeleteFront: function () {
        if (this.front == null) {
            if (this.rear == null) {
                alert("Список пуст!");
                return;
            } else {
                delete this.rear;
            }
        } else {
            if (this.count === 2) {
                this.front = this.rear;
                this.front.left = null;
                delete this.rear;
            } else {
                this.front = this.front.right;
                this.front.left = null;
            }
        }
        this.count--;
    },

    AddByIndex: function (number, data) {
        if (number < 0 || number > this.count) {
            alert("Invalid index!");
            return;
        } else {
            switch (this.count) {
                case 0:
                    this.front = new Node(undefined, undefined, data);
                    break;
                case 1:
                    if (number === 0) {
                        this.rear = this.front;
                        this.front = new Node(undefined, this.rear, data);
                        this.rear.left = this.front;
                    } else {
                        this.rear = new Node(this.front, undefined, data);
                        this.front.right = this.rear;
                    }
                    break;
                case 2:
                    if (number === 0) {
                        //this.rear = this.front;
```

```
                            this.front = new Node(undefined, this.front, data);
                            this.front.right.left = this.front;
                        } else if (number === 1) {
                            this.rear = new Node(this.front, this.rear, data);
                            this.rear.right.left = this.rear;
                        } else {
                            this.rear.right = new Node(this.rear, undefined, data);
                        }
                        break;
                    default:
                        if (number === 0) {
                            this.front.left = new Node(undefined, this.front,
data);
                            this.front = this.front.left;
                        } else {
                            let current = this.front;
                            for (let i = 0; i < number; i++) {
                                current = current.right;
                            }
                            current.left.right = new Node(current.left, current,
data);
                            current.left = current.left.right;
                        }
                }
            }
            this.count++;
        },

        DeleteByIndex: function (index) {
            if (index < 0 || index >= this.count) {
                alert("Invalid index!");
                return;
            }
            switch (this.count) {
                case 1:
                    delete this.front;
                    break;
                case 2:
                    switch (index) {
                        case 0:
                            this.front = this.back;
                            delete this.back;
                            break;
                        case 1:
                            delete this.back;
                            break;
                    }
                    break;
                default:
                    if (index === 0) {
                        this.DeleteFront();
                    } else if (index === this.count - 1) {
                        this.DeleteBack();
                    } else {
                        let current = this.front;
                        for (let i = 0; i < index; i++) {
                            current = current.right;
                        }
                        current.left.right = current.right;
                        current.right.left = current.left;
                    }

            }
            this.count--;
        },

        Draw: function () {
```

```
                var canvas = document.getElementById('canvas');
                2
                var ctx = canvas.getContext('2d');
                ctx.clearRect(0, 0, canvas.width, canvas.height);
                let current = this.front;
                for (let i = 0; i < this.count; i++) {
                    ctx.strokeRect(this.space * (i + 1) + this.width * i, 150,
this.width, this.height);
                    ctx.fillText(current.data, this.space * (i + 1) + this.width * i,
this.width + 95);
                    current = current.right;
                }
            }
        }
    </script>
</head>
<body>
<div style="text-align: center;">
    <canvas id="canvas" width="800" height="600"></canvas>
</div>
<div style="text-align: center;">
    <input type="button" value="Открыть меню" onclick="menu();"/>
</div>
<script>

    document.getElementById('canvas').getContext('2d').font = '48px serif';

    let menu = function () {
        const choice = prompt("1 - добавить элемент в начало, 2 - добавить элемент в
конец, 3 - добавить элемент по индексу, " +
            "4 - удалить элемент из начала, 5 - удалить элемент с конца, 6 удалить
элемент по индексу");

        switch (choice) {
            case "1":
                DoubleLinkedList.AddFront(parseInt(prompt("Введите число")));
                DoubleLinkedList.Draw();
                break;
            case "2":
                DoubleLinkedList.AddRear(parseInt(prompt("Введите число")));
                DoubleLinkedList.Draw();
                break;
            case "3":
                DoubleLinkedList.AddByIndex(parseInt(prompt("Введите индекс")),
parseInt(prompt("Введите число")));
                DoubleLinkedList.Draw();
                break;
            case "4":
                DoubleLinkedList.DeleteFront();
                DoubleLinkedList.Draw();
                break;
            case "5":
                DoubleLinkedList.DeleteBack();
                DoubleLinkedList.Draw();
                break;
            case "6":
                DoubleLinkedList.DeleteByIndex(parseInt(prompt("Введите индекс")));
                DoubleLinkedList.Draw();
                break;
        }
    }

</script>
</body>
</html>
```

Результат выполнения программы: