**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет Информационных технологий**

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № \_5\_**

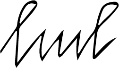
**Дисциплина:** *Технологии кроссплатформенного программирования*

**Тема:** *Объекты в JS*

**Выполнила: студентка группы \_\_\_***211-727***\_\_\_**

\_\_\_\_\_\_*Шемарева Анастасия Михайловна*\_\_\_\_\_\_\_

(Фамилия И.О.)



**Дата, подпись** \_\_\_\_*10.03.2023*\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2023**

**Лабораторная работа №5**

**«Объекты в JS»**

**Тема работы:** изучение приемов работы с объектами в JS.

**Цель работы:** изучить приёмы создания и использования объектов в языке JS

**Листинг программ**

**01.html**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

    <head>

        <title>ЛР5 Задание 1</title>

        <meta charset="utf-8"> <!-- для корректного вывода русского языка -->

    </head>

    <body>

        <script type="text/javascript">

            /\*

            Создать программную модель стека, используя объектно-ориентированный подход

            \*/

            var stack = {

                data: new Array,

                length: 0,

                maxsize: 1,

                pop() {

                    if (this.length > 0) {

                        let item = this.data.pop();

                        this.length--;

                        return item;

                    }

                    return "Пусто";

                },

                push(item) {

                    if (this.length < this.maxsize) {

                        this.data.push(item);

                        this.length++;

                        return 1;

                    }

                    return 0;

                }

            }

            // на основе аналогичного задания по ОСА

            var size = parseInt(prompt("Вы давно не разбирали свою личную библиотеку и решили составить башню из книг, чтобы потом рассортировать их." + "\nВ башне книги лежат друг на друге, и, чтобы снять определённую книгу, нужно сначала снять все книги над ней."+"\n\nСколько максимально книг вы планируете положить в башню?")); // получение размера стека

            if (size <= 0) alert("Вы передумали перебирать книги. В конце концов, можно заняться этим завтра.");

            else if (size > 0) {

                stack.maxsize = size; // задание стека

                var choice, book; // переменные для ответов пользователя

                do {

                    choice = parseInt(prompt("Вы решили, что будете работать с башней максимум из " + size + " книг(и)." + "\nТекущая башня: " + stack.data + ". Размер башни: " + stack.length + "\n\nДля работы с башней доступны следующие команды: " + "\n 0 - добавить книгу в башню;" + "\n 1 - забрать книгу из башни;" + "\n 2 - завершить работу: "));

                    switch(choice) {

                        case 0:

                            if (stack.length < size) {

                                book = prompt("Вы решили положить книгу в башню." + "\n\nВведите название книги, которую хотите положить в башню: ", "Книга " + Math.floor(Math.random() \* 100));

                                stack.push(book);

                                alert("Вы аккуратно кладёте книгу с названием \"" + book + "\" в башню.");

                            }

                            else alert("Вы решили положить книгу в башню." + "\nК сожалению, ваша башня уже слишком высокая, и вы откладываете книгу в сторону.");

                            break;

                        case 1:

                            if (stack.length > 0) {

                                book = stack.pop();

                                alert("Вы решили забрать книгу с башни." + "\nСамая верхняя книга - \"" + book + "\". Вы снимаете её с башни и откладываете в сторону.");

                            }

                            else alert("Вы решили забрать книгу с башни." + "\nК сожалению (или к радости) вы уже разобрали всю башню и в ней не осталось книг.");

                            break;

                        default:

                            break;

                    }

                }

                while (choice != 2);

                alert("Вы решили, что достаточно потрудились на сегодня, и отправились отдыхать.");

            }

        </script>

    </body>

</html>

**01.html (вариант с классом)**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

    <head>

        <title>ЛР5 Задание 1</title>

        <meta charset="utf-8"> <!-- для корректного вывода русского языка -->

    </head>

    <body>

        <script type="text/javascript">

            /\*

            Создать программную модель стека, используя объектно-ориентированный подход

            \*/

            class Stack {

                // соглашение программистов js - приватные поля называть с \_

                // не так давно были добавлены защищённые поля чрез #, но они не наследуются, поэтому не совсем подойдут в данном случае

                \_data = new Array;

                \_length = 0;

                \_maxSize = 1;

                constructor(size = 1) {

                    this.\_data = new Array;

                    this.\_length = 0;

                    this.\_maxsize = size;

                }

                // геттеры без сеттеров (аля приватные поля)

                // к ним всё ещё можно обратиться напрямую...

                get length() {

                    return this.\_length;

                }

                get maxsize() {

                    return this.\_maxsize;

                }

                get data() {

                    return this.\_data;

                }

                push(item) {

                    if (this.length < this.maxsize) {

                        this.data.push(item);

                        this.\_length++;

                        return 1;

                    }

                    return 0;

                }

                pop() {

                    if (this.length > 0) {

                        let item = this.data.pop();

                        this.\_length--;

                        return item;

                    }

                    return "Пусто";

                }

            }

            // на основе аналогичного задания по ОСА

            var size = parseInt(prompt("Вы давно не разбирали свою личную библиотеку и решили составить башню из книг, чтобы потом рассортировать их." + "\nВ башне книги лежат друг на друге, и, чтобы снять определённую книгу, нужно сначала снять все книги над ней."+"\n\nСколько максимально книг вы планируете положить в башню?")); // получение размера стека

            if (size <= 0) alert("Вы передумали перебирать книги. В конце концов, можно заняться этим завтра.");

            else if (size > 0) {

                var stack = new Stack(size); // задание стека

                var choice, book; // переменные для ответов пользователя

                //stack.\_length = 10; // можно менять "приватные" поля напрямую

                do {

                    choice = parseInt(prompt("Вы решили, что будете работать с башней максимум из " + size + " книг(и)." + "\nТекущая башня: " + stack.data + ". Размер башни: " + stack.length + "\n\nДля работы с башней доступны следующие команды: " + "\n 0 - добавить книгу в башню;" + "\n 1 - забрать книгу из башни;" + "\n 2 - завершить работу: "));

                    switch(choice) {

                        case 0:

                            if (stack.length < size) {

                                book = prompt("Вы решили положить книгу в башню." + "\n\nВведите название книги, которую хотите положить в башню: ", "Книга " + Math.floor(Math.random() \* 100));

                                stack.push(book);

                                alert("Вы аккуратно кладёте книгу с названием \"" + book + "\" в башню.");

                            }

                            else alert("Вы решили положить книгу в башню." + "\nК сожалению, ваша башня уже слишком высокая, и вы откладываете книгу в сторону.");

                            break;

                        case 1:

                            if (stack.length > 0) {

                                book = stack.pop();

                                alert("Вы решили забрать книгу с башни." + "\nСамая верхняя книга - \"" + book + "\". Вы снимаете её с башни и откладываете в сторону.");

                            }

                            else alert("Вы решили забрать книгу с башни." + "\nК сожалению (или к радости) вы уже разобрали всю башню и в ней не осталось книг.");

                            break;

                        default:

                            break;

                    }

                }

                while (choice != 2);

                alert("Вы решили, что достаточно потрудились на сегодня, и отправились отдыхать.");

            }

        </script>

    </body>

</html>

**02.html**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

    <head>

        <title>ЛР5 Задание 2</title>

        <meta charset="utf-8"> <!-- для корректного вывода русского языка -->

    </head>

    <body>

        <script type="text/javascript">

            /\*

            Создать программную модель циклической очереди, используя объектно-ориентированный подход

            \*/

            var queue = {

                data: new Array,

                head: 0,

                tail: 0,

                length: 0,

                maxsize: 1,

                push(item) {

                    this.tail %= this.maxsize;

                    if (this.length < this.maxsize) {

                        this.data[this.tail] = item;

                        this.tail++;

                        this.length++;

                        return 1;

                    }

                    return 0;

                },

                shift() {

                    this.head %= this.maxsize;

                    if (this.length > 0) {

                        let item = this.data[this.head];

                        this.data[this.head] = null;

                        this.head++;

                        this.length--;

                        return item;

                    }

                    return "Пусто";

                },

            }

            // на основе аналогичного задания по ОСА

            var size = parseInt(prompt("Вы давно не разбирали свою личную библиотеку и решили рассортировать книги на круглой полке после их успешной сборки в башни." + "\nНа полку книги кладутся по часовой стрелке. Для удобства на первой книге всегда отметка, которая передвигается по часовой стрелке к следующей книге, если текущую книгу с отметкой сняли." + " Вы всегда снимаете только отмеченную книгу, чтобы не потерять начало." + "\n\nСколько максимально книг вы планируете поставить на полку?")); // получение размера очереди

            if (size <= 0) alert("Вы передумали сортировать книги. В конце концов, можно заняться этим завтра.");

            else if (size > 0) {

                queue.maxsize = size;

                var choice, book;

                do {

                    choice = parseInt(prompt("Вы решили, что будете работать с полкой максимум из " + size + " книг(и)." + "\nТекущая полка: " + queue.data + ". Размер полки: " + queue.length + "\n\nДля работы с полкой доступны следующие команды: " + "\n 0 - добавить книгу на полку;" + "\n 1 - забрать книгу с полки;" + "\n 2 - завершить работу: "));

                    switch(choice) {

                        case 0:

                            if (queue.length < size) {

                                book = prompt("Вы решили поставить книгу на полку." + "\n\nВведите название книги, которую хотите поставить на полу: ", "Книга " + Math.floor(Math.random() \* 100));

                                queue.push(book);

                                alert("Вы аккуратно ставите книгу с названием \"" + book + "\" на полку.");

                            }

                            else alert("Вы решили поставить книгу на полку." + "\nК сожалению, лимит книг на полке уже достигнут, и вы откладываете книгу в сторону.");

                            break;

                        case 1:

                            if (queue.length > 0) {

                                book = queue.shift();

                                alert("Самая первая книга - \"" + book + "\". Вы снимаете её с полки и откладываете в сторону.");

                            }

                            else alert("К сожалению (или к радости) вы уже разобрали всю полку и в ней не осталось книг.");

                            break;

                        default:

                            break;

                    }

                }

                while (choice != 2);

                alert("Вы решили, что достаточно потрудились на сегодня, и отправились отдыхать."); // сообщение о завершении работы

            }

        </script>

    </body>

</html>

**03.html**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

    <head>

        <title>ЛР5 Задание 3</title>

        <meta charset="utf-8"> <!-- для корректного вывода русского языка -->

    </head>

    <body>

        <script type="text/javascript">

            /\*

            Создать программную модель двусвязного списка, используя объектно-ориентированный подход

            \*/

            // на основании аналогичного задания по оса

            class Node {

                constructor(value) {

                    this.value = value;

                    this.prev = null;

                    this.next = null;

                }

            }

            var List = {

                top: null,

                root: null,

                nodeCount: 0,

                addTopNode(value) {

                    let nd = new Node(value);

                    if (this.nodeCount == 0) {

                        this.root = nd;

                        this.top = nd;

                        this.nodeCount++;

                        return "Элемент " + value + " добавлен в качестве первого элемента";

                    }

                    else {

                        this.top.next = nd;

                        nd.prev = this.top;

                        this.top = nd;

                        this.nodeCount++;

                        return "Элемент " + value + " добавлен в конец";

                    }

                },

                addRootNode(value) {

                    let nd = new Node(value);

                    if (this.nodeCount == 0) {

                        this.root = nd;

                        this.top = nd;

                        this.nodeCount++;

                        return "Элемент " + value + " добавлен в качестве первого элемента";

                    }

                    else {

                        nd.next = this.root;

                        this.root.prev = nd;

                        this.root = nd;

                        this.nodeCount++;

                        return "Элемент " + value + " добавлен в начало";

                    }

                },

                removeTopNode() {

                    if (this.top === null) return "Список пуст";

                    else if (this.root == this.top) return this.removeRootNode();

                    let nd = this.top.prev;

                    nd.next = null;

                    delete this.top;

                    this.top = nd;

                    this.nodeCount--;

                    if (this.nodeCount == 0) {

                        this.top = null;

                        this.root = null;

                    }

                    else if (this.nodeCount == 1) this.top = this.root;

                    return "Конечный элемент удалён";

                },

                removeRootNode() {

                    if (this.root == null) return "Список пуст";

                    let nd = this.root;

                    this.root = nd.next;

                    nd.prev = null;

                    delete nd;

                    this.nodeCount--;

                    if (this.nodeCount == 0) {

                        this.top = null;

                        this.root = null;

                    }

                    else if (this.nodeCount == 1) this.top = this.root;

                    return "Начальный элемент удалён";

                },

                findNode(value) {

                    if (this.nodeCount > 0) {

                        let nd = this.root;

                        for (let i = 0; i < this.nodeCount; i++) {

                            if (nd.value == value) return "Элемент " + nd.value + " найден на позиции " + i;

                            nd = nd.next;

                        }

                        return "Элемент не найден";

                    }

                    return "Список пуст";

                },

                removeFoundNode(foundvalue) {

                    let foundnode = this.root;

                    while (foundnode.value != foundvalue) foundnode = foundnode.next;

                    if (this.root == foundnode) return this.removeRootNode();

                    else if (this.top == foundnode) return this.removeTopNode();

                    else {

                        foundnode.prev.next = foundnode.next;

                        foundnode.next.prev = foundnode.prev;

                        delete foundnode;

                        this.nodeCount--;

                        return "Найденный элемент " + foundvalue + " удалён";

                    }

                },

                addAfterFoundNode(foundvalue, newvalue) {

                    let foundnode = this.root;

                    let addnode = new Node(newvalue);

                    while (foundnode.value != foundvalue) foundnode = foundnode.next;

                    if (this.top == foundnode) return this.addTopNode(newvalue);

                    addnode.next = foundnode.next;

                    addnode.prev = foundnode;

                    foundnode.next = addnode;

                    this.nodeCount++;

                    return "Элемент " + newvalue + " добавлен после элемента " + foundvalue;

                },

                print() {

                    let list = "";

                    if (this.nodeCount > 0) {

                        let nd = this.root;

                        for (let i = 0; i < this.nodeCount; i++) {

                            list += nd.value;

                            if (nd.next != null) {

                                list += ", ";

                            }

                            nd = nd.next;

                        }

                    }

                    return list;

                }

            }

            var choice, item;

            do {

                choice = parseInt(prompt("Текущий список: " + List.print() + "\n\nДля работы со списком доступны следующие команды: " + "\n 1 - добавить элемент в начало;" + "\n 2 - добавить элемент в конец;" + "\n 3 - удалить начальный элемент;" + "\n 4 - удалить конечный элемент;" + "\n 5 - найти элемент по значению;" + "\n 0 - завершить работу: "));

                switch(choice) {

                    case 1:

                        item = prompt("Добавление элемента в начало." + "\n\nВведите значение элемента, который хотите добавить: ", Math.floor(Math.random() \* 100));

                        alert(List.addRootNode(item));

                        break;

                    case 2:

                        item = prompt("Добавление элемента в конец." + "\n\nВведите значение элемента, который хотите добавить: ", Math.floor(Math.random() \* 100));

                        alert(List.addTopNode(item));

                        break;

                    case 3:

                        alert(List.removeRootNode());

                        break;

                    case 4:

                        alert(List.removeTopNode());

                        break;

                    case 5:

                        item = prompt("Поиск элемента по значению." + "\n\nВведите значение искомого элемента: ");

                        if (List.findNode(item) != "Элемент не найден" && List.findNode(item) != "Список пуст") {

                            do {

                                choice = parseInt(prompt(List.findNode(item) + "\n\nВы можете совершить следующие действия: " + "\n 1 - удалить найденный элемент;" + "\n 2 - добавить новый элемент после найденного;" + "\n 3 - вернуться к списку основных действий;" + "\n 0 - завершить работу: "));

                            }

                            while (choice < 0 && choice > 3);

                            if (choice == 1) {

                                alert(List.removeFoundNode(item));

                            }

                            else if (choice == 2) {

                                let newitem = prompt("Добавление элемента после найденного." + "\n\nВведите значение элемента, который хотите добавить: ", Math.floor(Math.random() \* 100));

                                alert(List.addAfterFoundNode(item, newitem));

                            }

                        }

                        else alert(List.findNode(item));

                        break;

                    default:

                        break;

                }

            }

            while (choice != 0);

            alert("Завершение работы");

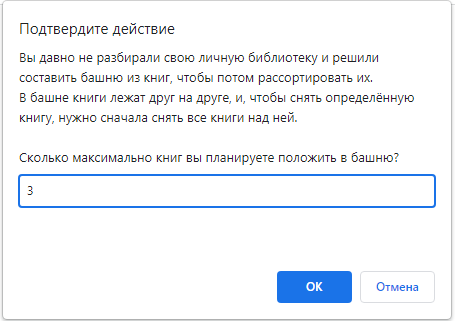
        </script>

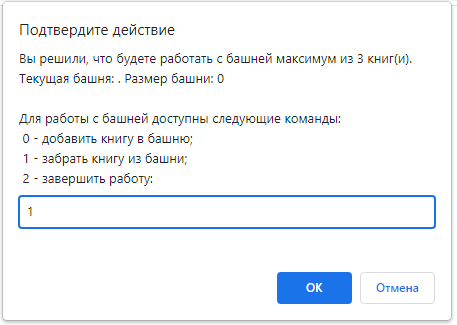
    </body>

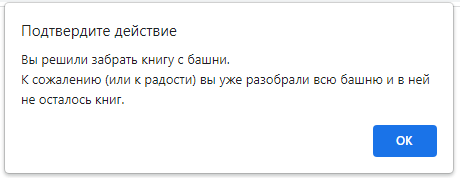
</html>

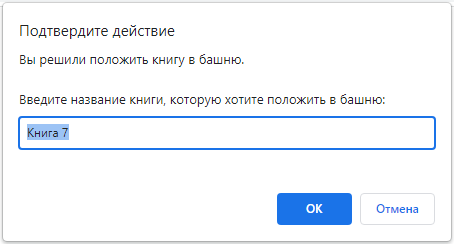
**Скриншоты работы программ**

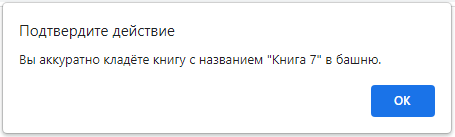
**01.html**

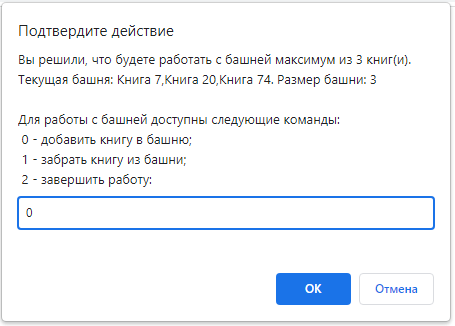


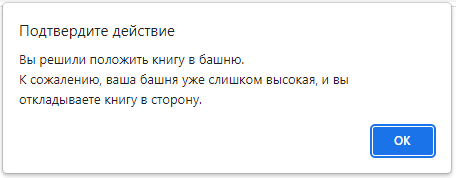


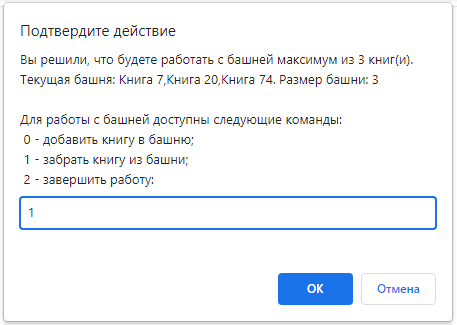


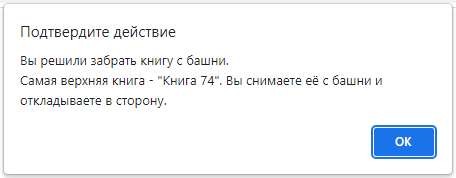


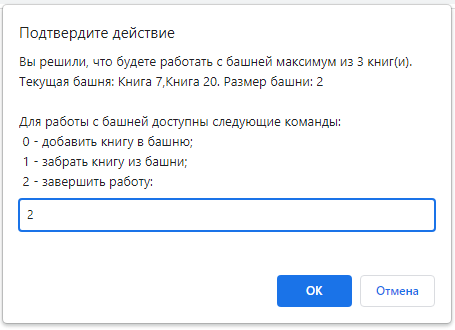


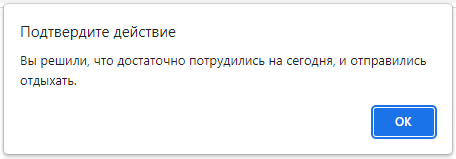




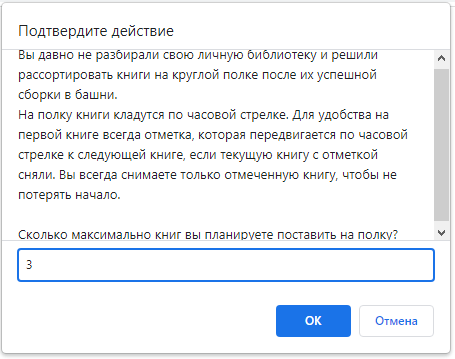


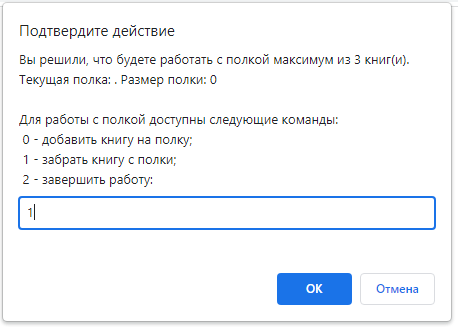


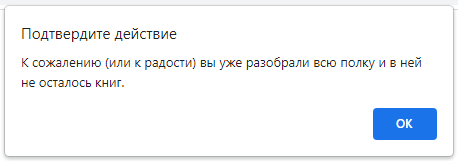


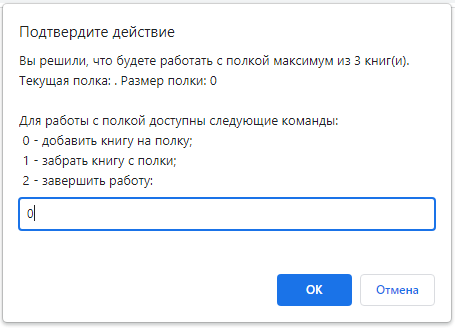


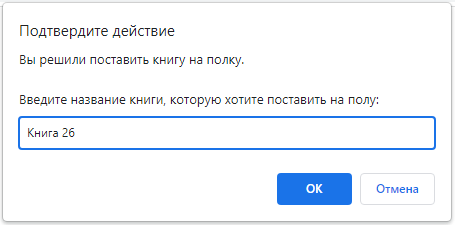
**02.html**

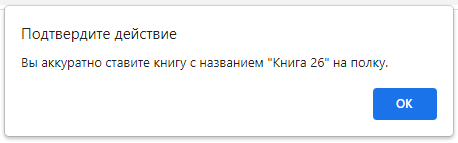


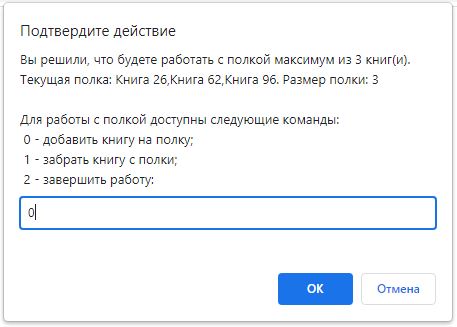


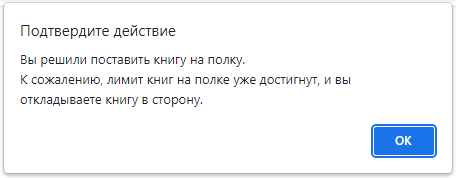


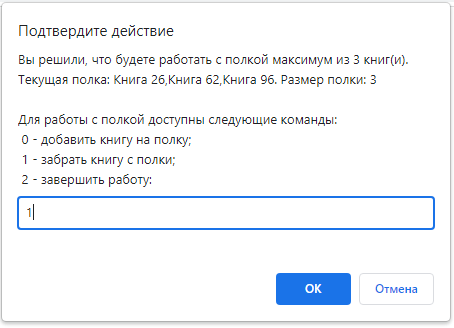


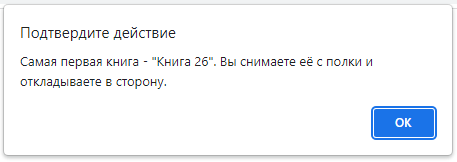


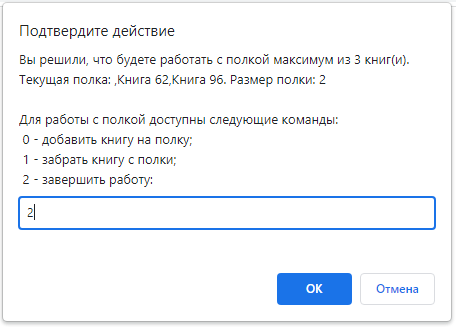


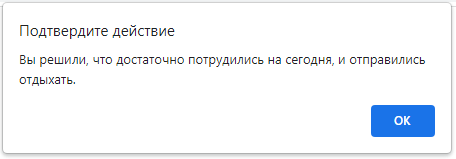












**03.html**

