**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 № \_6\_**

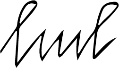
**Дисциплина:** *Технологии кроссплатформенного программирования*

**Тема:** *Интерактивность в JS*

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

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

(Фамилия И.О.)



**Дата, подпись** \_\_\_\_*18.03.2023*\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

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

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

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

(Дата) (Подпись)

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

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

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

**Москва**

**2023**

**Лабораторная работа №5**

**«Интерактивность в JS»**

**Тема работы:** изучение приемов создания интерактивных приложений в JS.

**Цель работы:** изучить приёмы создания интерактивных приложений с использованием Canvas 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>ЛР6 Задание 1</title>

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

        <style>

            p#err {

                color: red

            }

            p#log {

                max-height: 500px;

                overflow-y: scroll;

                border: 2px dashed black;

                padding: 5px;

            }

            #pop, #log {

                visibility: hidden;

            }

        </style>

    </head>

    <body>

        <form id="frm" autocomplete="off">

            <input type="text" id="el" placeholder="Размер башни (стека)" maxlength="2" pattern="[0-9]\*" autofocus>

            <input type="submit" id="push" value="Применить">

            <input type="button" id="pop" value="Снять последнюю книгу с башни">

        </form>

        <br>

        <canvas id="canv" width="800px"></canvas>

        <br>

        <p id="err">

        <p id="info">Вы давно не разбирали свою личную библиотеку и решили составить башню из книг, чтобы потом рассортировать их.<br>

            В башне книги лежат друг на друге, и, чтобы снять определённую книгу, нужно сначала снять все книги над ней.<br>

            Сколько максимально книг вы планируете положить в башню?

        <p id="log">

        </p>

        <script type="text/javascript">

            /\*

            Для объектно-ориентированной программной модели стека разработать пользовательский графический

            интерфейс.

            \*/

            var canvas = document.getElementById("canv");

            var ctx = canvas.getContext('2d');

            var el = document.getElementById("el");

            var btnPush = document.getElementById("push");

            var btnPop = document.getElementById("pop");

            var err = document.getElementById("err");

            var info = document.getElementById("info");

            var log = document.getElementById("log");

            var frm = document.getElementById("frm");

            frm.addEventListener("submit", function(e){

                e.preventDefault();

            })

            btnPush.addEventListener("click", sizeCheck);

            function sizeCheck() {

                if (el.value <= 0) err.innerHTML = "Высота башни должна быть больше нуля";

                else if (!el.checkValidity())  err.innerHTML = "Высота башни должна быть числом";

                else {

                    stack.maxsize = +el.value;

                    err.innerHTML = "";

                    btnPush.removeEventListener("click", sizeCheck);

                    btnPush.addEventListener("click", push);

                    btnPush.value = "Положить в башню";

                    el.value = "";

                    el.placeholder = "Название книги";

                    el.pattern = "\*";

                    el.maxLength = 20;

                    btnPop.style.visibility = "visible";

                    btnPop.addEventListener("click", pop);

                    log.style.visibility = "visible";

                    reloadInfo();

                    drawStack();

                }

            }

            function reloadInfo() {

                info.innerHTML = "Вы решили, что будете работать с башней максимум из " + stack.maxsize + " книг(и)." + "<br>Текущая высота башни: " + stack.length + "<br>Для удобства верхняя книга отмечена зелёным цветом.";

            }

            function push(item) {

                item = el.value;

                if (item == "") err.innerHTML = "Введите название книги";

                else {

                    let p = stack.push(item);

                    if (p) {

                        log.innerHTML += "Вы решили положить книгу в башню." + "<br>" + "Вы аккуратно кладёте книгу с названием \"" + item + "\" в башню.";

                        el.value = err.innerHTML = "";

                        drawStack();

                    }

                    else {

                        log.innerHTML += "Вы решили положить книгу в башню." + "<br>" + "К сожалению, ваша башня уже слишком высокая, и вы откладываете книгу в сторону.";

                        err.innerHTML = "Стак переполнен";

                    }

                    updLog();

                }

            }

            function pop() {

                let p = stack.pop();

                if (p != "Пусто") {

                    log.innerHTML += "Вы решили забрать книгу с башни." + "<br>" + "Самая верхняя книга - \"" + p + "\"." + "<br>" + "Вы снимаете её с башни и откладываете в сторону.";

                    err.innerHTML = "";

                    drawStack();

                }

                else {

                    log.innerHTML += "Вы решили забрать книгу с башни." + "<br>" + "К сожалению (или к радости), вы уже разобрали всю башню и в ней не осталось книг.";

                    err.innerHTML = "Стак пуст";

                }

                updLog();

            }

            function updLog() {

                log.innerHTML += "<br><br>";

                log.scrollTo(0, log.scrollHeight);

                reloadInfo();

            }

            function drawStack() {

                let font = 20;

                let height = font + 4;

                ctx.clearRect(0, 0, canvas.width, canvas.height);

                canvas.height = stack.maxsize \* height + 10;

                ctx.font = font+'px Arial';

                ctx.textAlign = "center";

                for (let i = 1; i < stack.length + 1; i++) {

                    let rand = Math.random() \* 30 + 100;

                    if (i == stack.length) ctx.fillStyle = "lime";

                    else ctx.fillStyle = "white";

                    ctx.fillRect(rand, canvas.height - (i-1) \* height - height, stack.data[i-1].toString().length \* font, height);

                    ctx.strokeRect(rand, canvas.height - (i-1) \* height - height, stack.data[i-1].toString().length \* font, height);

                    ctx.fillStyle = "black";

                    ctx.fillText(stack.data[i-1], stack.data[i-1].toString().length \* 10 + rand, canvas.height - (i-1) \* height - 4);

                }

            }

            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 true;

                    }

                    return false;

                }

            }

        </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>ЛР6 Задание 2</title>

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

        <style>

            p#err {

                color: red

            }

            p#log {

                max-height: 500px;

                overflow-y: scroll;

                border: 2px dashed black;

                padding: 5px;

            }

            #pop, #log {

                visibility: hidden;

            }

        </style>

    </head>

    <body>

        <form id="frm" autocomplete="off">

            <input type="text" id="el" placeholder="Размер полки (очереди)" maxlength="2" pattern="[0-9]\*">

            <input type="submit" id="push" value="Применить">

            <input type="button" id="pop" value="Снять отмеченную книгу с полки">

        </form>

        <br>

        <canvas id="canv" width="800px"></canvas>

        <br>

        <p id="err">

        <p id="info">Вы давно не разбирали свою личную библиотеку и решили рассортировать книги на круглой полке после их успешной сборки в башни.<br>

            На 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.<br>

            Вы всегда снимаете только отмеченную книгу, чтобы не потерять начало.<br>

            Сколько максимально книг вы планируете поставить на полку?

        <p id="log">

        </p>

        <script type="text/javascript">

            /\*

            Для объектно-ориентированной программной модели циклической очереди разработать

            пользовательский графический интерфейс.

            \*/

            var canvas = document.getElementById("canv");

            var ctx = canvas.getContext('2d');

            var el = document.getElementById("el");

            var btnPush = document.getElementById("push");

            var btnPop = document.getElementById("pop");

            var err = document.getElementById("err");

            var info = document.getElementById("info");

            var log = document.getElementById("log");

            // чтобы можно было отправлять на enter, но страница не обновлялась

            var frm = document.getElementById("frm");

            frm.addEventListener("submit", function(e){

                e.preventDefault();

            })

            btnPush.addEventListener("click", sizeCheck);

            function sizeCheck() {

                if (el.value <= 0) err.innerHTML = "Размер полки делжен быть больше нуля";

                else if (!el.checkValidity())  err.innerHTML = "Размер полки должент быть числом";

                else {

                    queue.maxsize = +el.value;

                    err.innerHTML = "";

                    btnPush.removeEventListener("click", sizeCheck);

                    btnPush.addEventListener("click", push);

                    btnPush.value = "Поставить на полку";

                    el.value = "";

                    el.placeholder = "Название книги";

                    el.pattern = "\*";

                    el.maxLength = 20;

                    btnPop.style.visibility = "visible";

                    btnPop.addEventListener("click", shift);

                    log.style.visibility = "visible";

                    reloadInfo();

                    drawQueue();

                }

            }

            function reloadInfo() {

                info.innerHTML = "Вы решили, что будете работать с полкой максимум из " + queue.maxsize + " книг(и)." + "<br>Текущий размер полки: " + queue.length + "<br>Для удобства первая книга (либоо место, на котором будет первая книга) отмечена зелёным цветом.";

            }

            function push(item) {

                item = el.value;

                if (item == "") err.innerHTML = "Введите название книги";

                else {

                    let p = queue.push(item);

                    if (p) {

                        log.innerHTML += "Вы решили поставить книгу на полку." + "<br>" + "Вы аккуратно ставите книгу с названием \"" + item + "\" на полку.";

                        el.value = err.innerHTML = "";

                        drawQueue();

                    }

                    else {

                        log.innerHTML += "Вы решили поставить книгу на полку." + "<br>" + "К сожалению, лимит книг на полке уже достигнут, и вы откладываете книгу в сторону.";

                        err.innerHTML = "Очередь переполнена";

                    }

                    updLog();

                }

            }

            function shift() {

                let p = queue.shift();

                if (p != "Пусто") {

                    log.innerHTML += "Вы решили забрать книгу с полки." + "<br>" + "Самая первая книга - \"" + p + "\"." + "<br>" + "Вы снимаете её с полки и откладываете в сторону.";

                    err.innerHTML = "";

                    drawQueue();

                }

                else {

                    log.innerHTML += "Вы решили забрать книгу с полки." + "<br>" + "К сожалению (или к радости), вы уже разобрали всю полку и на ней не осталось книг.";

                    err.innerHTML = "Очередь пуста";

                }

                updLog();

            }

            function updLog() {

                log.innerHTML += "<br><br>";

                log.scrollTo(0, log.scrollHeight);

                reloadInfo();

            }

            function drawQueue() {

                let font = 20;

                let height = font + 4;

                ctx.clearRect(0, 0, canvas.width, canvas.height);

                canvas.height = queue.maxsize \* height \* 2;

                ctx.font = font+'px Arial';

                var radius = canvas.height / 2 - height / 2;

                var ang;

                for (let i = 1; i < queue.maxsize + 1; i++) {

                    ang = i \* Math.PI / queue.maxsize \* 2;

                    ctx.rotate(ang);

                    ctx.translate(0, -radius\*0.85);

                    ctx.rotate(-ang);

                    if (i-1 == queue.head) ctx.fillStyle = "lime";

                    else ctx.fillStyle = "white";

                    ctx.fillRect(radius, radius, queue.data[i-1] == null? 10 : queue.data[i-1].toString().length \* height/2, height);

                    ctx.strokeRect(radius, radius, queue.data[i-1] == null? 10 : queue.data[i-1].toString().length \* height/2, height);

                    ctx.fillStyle = "black";

                    if (queue.data[i-1] != null) ctx.fillText(queue.data[i-1],  radius + queue.data[i-1].toString().length/2, radius + height - 4);

                    ctx.rotate(ang);

                    ctx.translate(0, radius\*0.85);

                    ctx.rotate(-ang);

                }

            }

            var queue = {

                data: new Array,

                head: 0,

                tail: 0,

                length: 0,

                maxsize: 1,

                shift() {

                    this.head %= this.maxsize;

                    if (this.length > 0) {

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

                        this.data[this.head] = null;

                        this.head++;

                        this.head %= this.maxsize;

                        this.length--;

                        return item;

                    }

                    return "Пусто";

                },

                push(item) {

                    this.tail %= this.maxsize;

                    if (this.length < this.maxsize) {

                        this.data[this.tail] = item;

                        this.tail++;

                        this.tail %= this.maxsize;

                        this.length++;

                        return true;

                    }

                    return false;

                }

            }

        </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>ЛР6 Задание 3</title>

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

        <style>

            p#err {

                color: red

            }

            p#log {

                max-height: 500px;

                overflow-y: scroll;

                border: 2px dashed black;

                padding: 5px;

            }

            .root {

                background-color: cadetblue;

            }

            .top {

                background-color: coral;

            }

            .found {

                background-color: orchid;

                visibility: hidden;

            }

        </style>

    </head>

    <body>

        <form id="frm1" autocomplete="off">

            <fieldset>

                <legend>Вставка</legend>

                <input type="text" id="el" placeholder="Данные для вставки" maxlength="20" autofocus>

                <input type="button" id="pushFound" class="found" value="Вставить после найденного элемента">

                <br>

                <input type="submit" id="pushRoot" class="root" value="Вставить в начало списка">

                <input type="button" id="pushTop" class="top" value="Вставить в конец списка">

            </fieldset>

        </form>

        <br>

        <form>

            <fieldset>

                <legend>Удаление</legend>

                <input type="button" id="popRoot" class="root" value="Удалить головной элемент">

                <input type="button" id="popTop" class="top" value="Удалить хвостовой элемент">

            </fieldset>

        </form>

        <br>

        <form id="frm2" autocomplete="off">

            <fieldset>

                <legend>Поиск</legend>

                <input type="text" id="elFind" placeholder="Данные для поиска" maxlength="20" title="Индекс должен быть целым числом >= 0">

                <input type="submit" id="find" value="Найти">

                <br>

                <label><input type="radio" name="findType" id="findItem" value="item">Поиск по значению</label>

                <label><input type="radio" name="findType" id="findIndex" value="index">Поиск по индексу</label>

                <br>

                <input type="button" id="popFound" class="found" value="Удалить найденный элемент">

            </fieldset>

        </form>

        <br>

        <canvas id="canv" width="800px"></canvas>

        <br>

        <p id="err">

        <p id="info">Вы давно не разбирали свою личную библиотеку и решили составить башню из книг, чтобы потом рассортировать их.<br>

            В башне книги лежат друг на друге, и, чтобы снять определённую книгу, нужно сначала снять все книги над ней.<br>

            Сколько максимально книг вы планируете положить в башню?

        <p id="log">

        </p>

        <script type="text/javascript">

            /\*

            Для объектно-ориентированной программной модели двусвязного списка разработать пользовательский

            графический интерфейс.

            \*/

            var canvas = document.getElementById("canv");

            var ctx = canvas.getContext('2d');

            var el = document.getElementById("el");

            var btnPushTop = document.getElementById("pushTop");

            var btnPushRoot = document.getElementById("pushRoot");

            var btnPopTop = document.getElementById("popTop");

            var btnPopRoot = document.getElementById("popRoot");

            var elFind = document.getElementById("elFind");

            var btnFind = document.getElementById("find");

            var btnPushFound = document.getElementById("pushFound");

            var btnPopFound = document.getElementById("popFound");

            var radItem = document.getElementById("findItem");

            var radIndex = document.getElementById("findIndex");

            var err = document.getElementById("err");

            var info = document.getElementById("info");

            var log = document.getElementById("log");

            var frm1 = document.getElementById("frm1");

            var frm2 = document.getElementById("frm2");

            frm1.addEventListener("submit", function(e){

                e.preventDefault();

            })

            frm2.addEventListener("submit", function(e){

                e.preventDefault();

            })

            btnPushTop.addEventListener("click", pushTop);

            btnPushRoot.addEventListener("click", pushRoot);

            btnPopTop.addEventListener("click", popTop);

            btnPopRoot.addEventListener("click", popRoot);

            btnFind.addEventListener("click", findNode);

            radItem.checked = true;

            radItem.addEventListener("click", function() {

                if (!radItem.checked) elFind.pattern = "[0-9]\*";

                else elFind.pattern = "\*";

            })

            radIndex.addEventListener("click", function() {

                if (radIndex.checked) elFind.pattern = "[0-9]\*";

                else elFind.pattern = "\*";

            })

            btnPushFound.addEventListener("click", addAfterFound);

            btnPopFound.addEventListener("click", removeFound);

            function reloadInfo() {

                info.innerHTML = "Текущая длина списка: " + List.nodeCount;

            }

            function pushTop(item) {

                item = el.value;

                if (item == "") err.innerHTML = "Элемент для вставки должен быть непустым";

                else {

                    err.innerHTML = "";

                    el.value = "";

                    let p = List.addTopNode(item);

                    log.innerHTML += p;

                    updLog();

                    drawList();

                    reloadInfo();

                }

            }

            function pushRoot(item) {

                item = el.value;

                if (item == "") err.innerHTML = "Элемент для вставки должен быть непустым";

                else {

                    err.innerHTML = "";

                    el.value = "";

                    let p = List.addRootNode(item);

                    log.innerHTML += p;

                    updLog();

                    drawList();

                    reloadInfo();

                }

            }

            function popRoot() {

                let p = List.removeRootNode();

                if (p == "Список пуст") err.innerHTML = p;

                else {

                    err.innerHTML = "";

                    log.innerHTML += p;

                    updLog();

                    drawList();

                    reloadInfo();

                }

            }

            function popTop() {

                let p = List.removeTopNode();

                if (p == "Список пуст") err.innerHTML = p;

                else {

                    err.innerHTML = "";

                    log.innerHTML += p;

                    updLog();

                    drawList();

                    reloadInfo();

                }

            }

            var foundInfo = "";

            function findNode(item) {

                item = elFind.value;

                if (item == "") err.innerHTML = "Элемент для поиска должен быть непустым";

                else if (radIndex.checked && (!elFind.checkValidity() || (elFind.checkValidity() && item < 0))) err.innerHTML = "Индекс для поиска должен быть числом >= 0";

                else {

                    foundInfo = (radItem.checked)? List.findNode(item) : List.findNode("", item);

                    if (typeof(foundInfo) === "string") err.innerHTML = foundInfo;

                    else {

                        log.innerHTML += foundInfo.text;

                        info.innerHTML = foundInfo.text;

                        err.innerHTML = "";

                        btnPushFound.style.visibility = btnPopFound.style.visibility = "visible";

                        updLog();

                        drawList();

                    }

                }

            }

            function removeFound(item) {

                if (foundInfo == "") err.innerHTML = "Поиск не был произведён";

                else {

                    item = foundInfo.item;

                    let p = List.removeFoundNode(item);

                    foundInfo = err.innerHTML = elFind.value = info.innerHTML = "";

                    btnPushFound.style.visibility = btnPopFound.style.visibility = "hidden";

                    log.innerHTML += p + "<br>Результаты поиска очищены";

                    updLog();

                    drawList();

                    reloadInfo();

                }

            }

            function addAfterFound(foundVal, newVal) {

                foundVal = foundInfo.item;

                newVal = el.value;

                if (el.value == "") err.innerHTML = "Элемент для вставки должен быть непустым";

                else {

                    let p = List.addAfterFoundNode(foundVal, newVal);

                    err.innerHTML = "";

                    el.value = "";

                    log.innerHTML += p;

                    drawList();

                    updLog();

                }

            }

            function updLog() {

                log.innerHTML += "<br><br>";

                log.scrollTo(0, log.scrollHeight);

            }

            function drawList() {

                let font = 20;

                let height = font + 4;

                canvas.width += List.getLength() \* font/2;

                ctx.clearRect(0, 0, canvas.width, canvas.height);

                ctx.font = font+'px Arial';

                ctx.textAlign = "center";

                let length = 0;

                for (let i = 1; i < List.nodeCount + 1; i++) {

                    if (foundInfo != "" && foundInfo.index == i-1) ctx.fillStyle = "orchid";

                    else if (i == 1) ctx.fillStyle = "cadetblue";

                    else if (i == List.nodeCount) ctx.fillStyle = "coral";

                    else ctx.fillStyle = "white";

                    let data = List.findNode("", i-1).item.toString();

                    ctx.fillRect(length, 50, data.length \* height, height);

                    ctx.strokeRect(length, 50, data.length \* height, height);

                    ctx.fillStyle = "black";

                    ctx.fillText(data, length + data.length \* height/2, 50 + height - 4);

                    if (i != List.nodeCount) ctx.fillText("<=>", length + data.length \* height + font, 50 + height - 4);

                    length += data.length \* height + 40;

                }

            }

            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;

                    let val = this.top.value;

                    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 " + val + " удалён";

                },

                removeRootNode() {

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

                    let nd = this.root;

                    let val = this.root.value;

                    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 " + val + " удалён";

                },

                findNode(value = "", index = -1) { // добавлен поиск по индексу

                    if (this.nodeCount > 0) {

                        if (index >= this.nodeCount) return "Заданный индекс >= числу элементов";

                        let nd = this.root;

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

                            if (index != -1 && i == index) return {

                                text: "На позиции " + i + " найден элемент "  + nd.value,

                                item: nd.value,

                                index: i,

                            };

                            else if (index == -1 && nd.value == value) return {

                                text: "Элемент " + nd.value + " найден на позиции " + i,

                                item: nd.value,

                                index: 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;

                },

                // новая функция для динамичной ширины canvas

                getLength() {

                    let length = 0;

                    let nd = this.root;

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

                        length += nd.value.toString().length;

                        nd = nd.next;

                    }

                    return length;

                }

            }

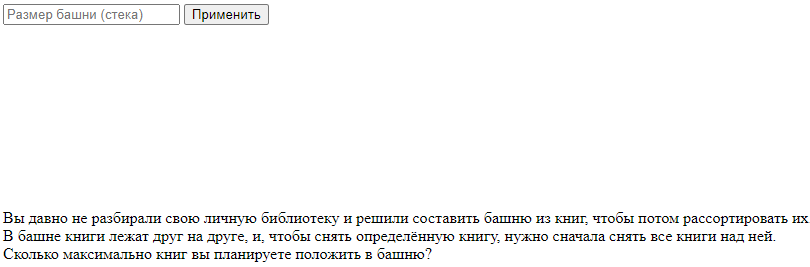
        </script>

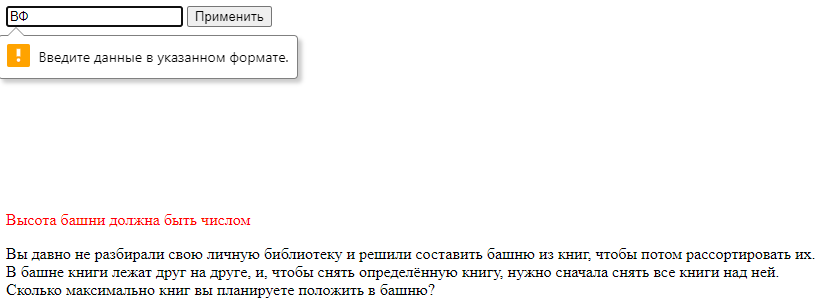
    </body>

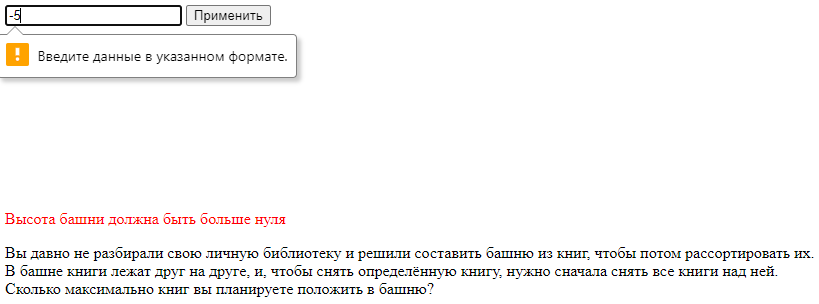
</html>

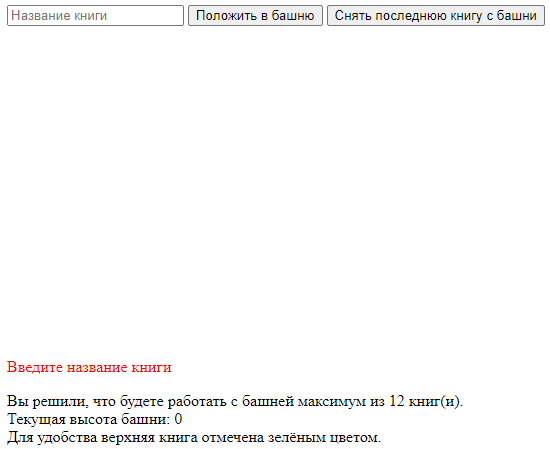
**Скриншоты работы программ**

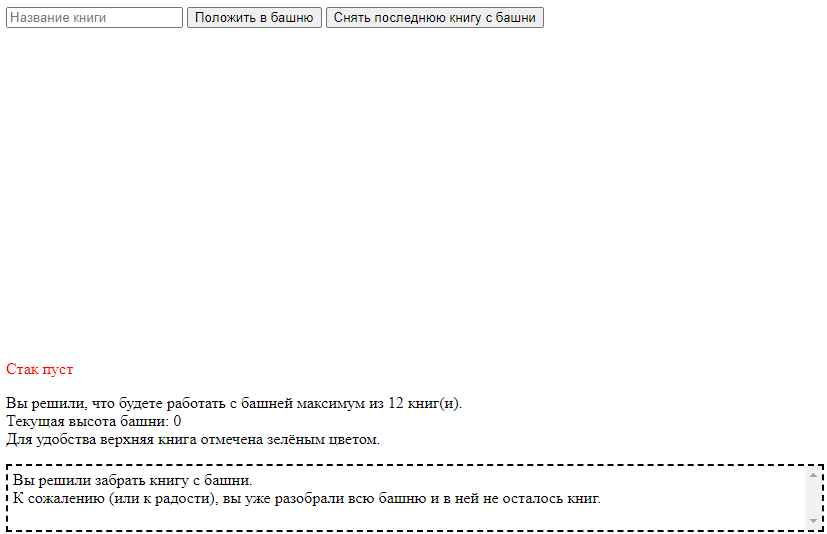
**01.html**





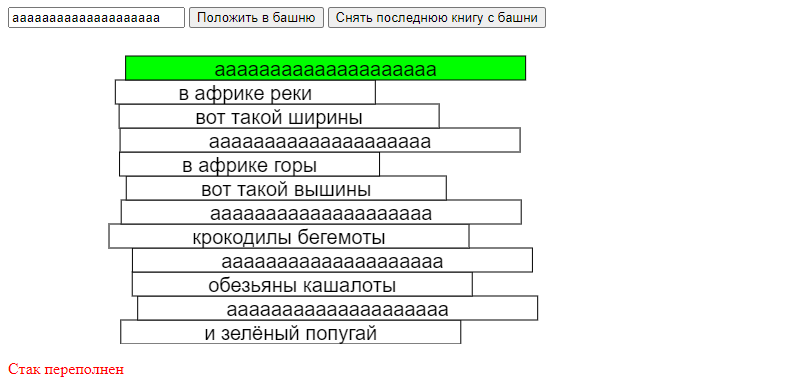


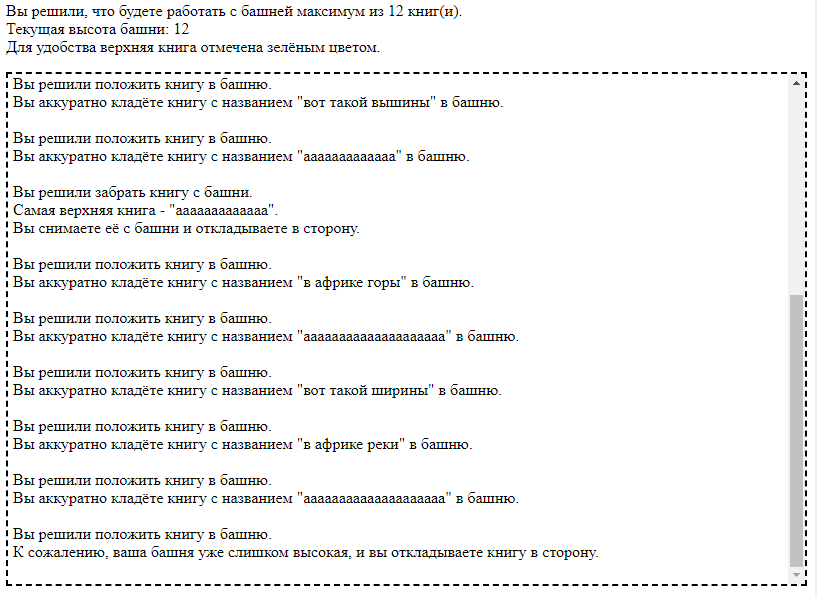




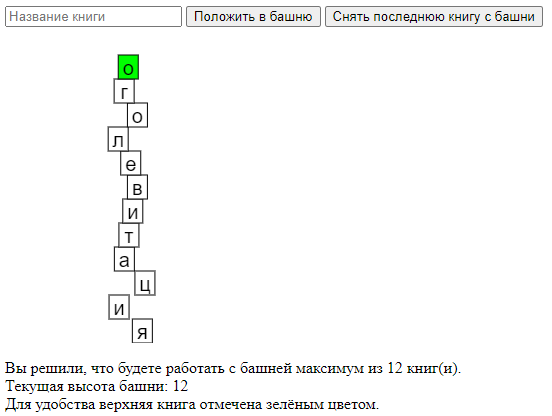




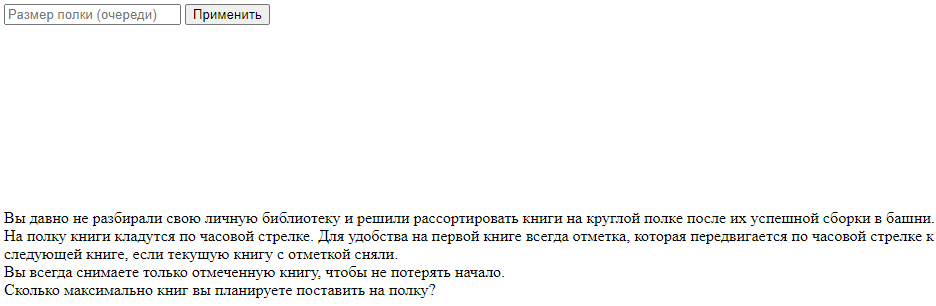


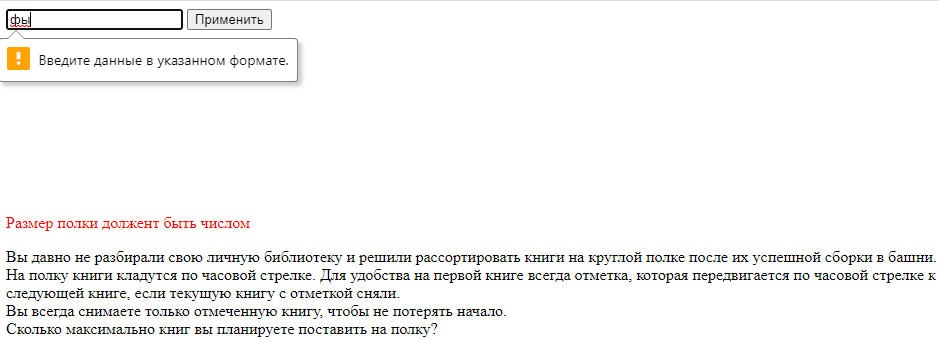


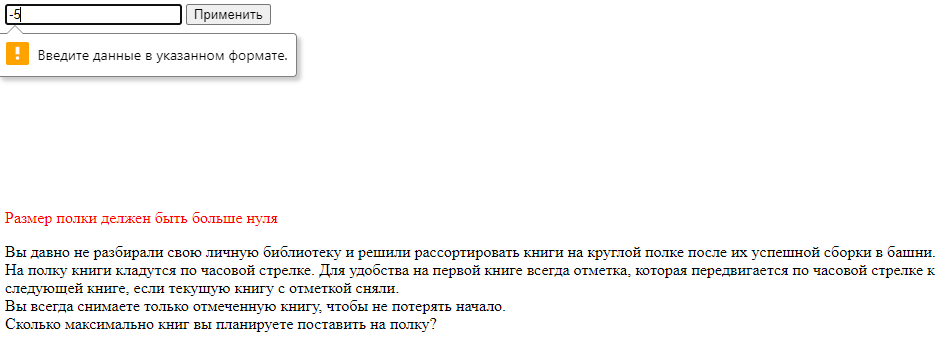
С маленькими названиями можно увидеть чудеса балансировки и левитации:



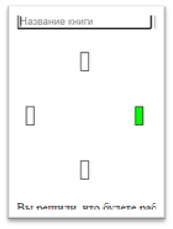
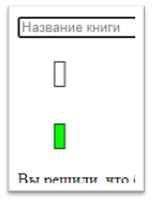
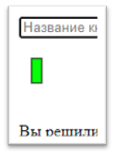
**02.html**

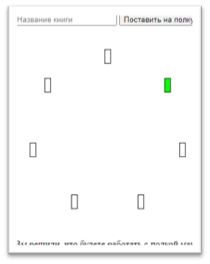
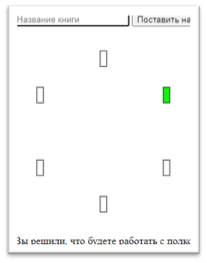
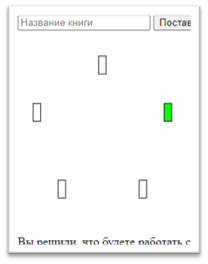




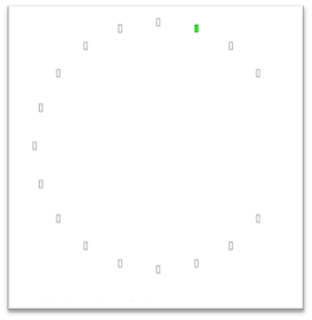


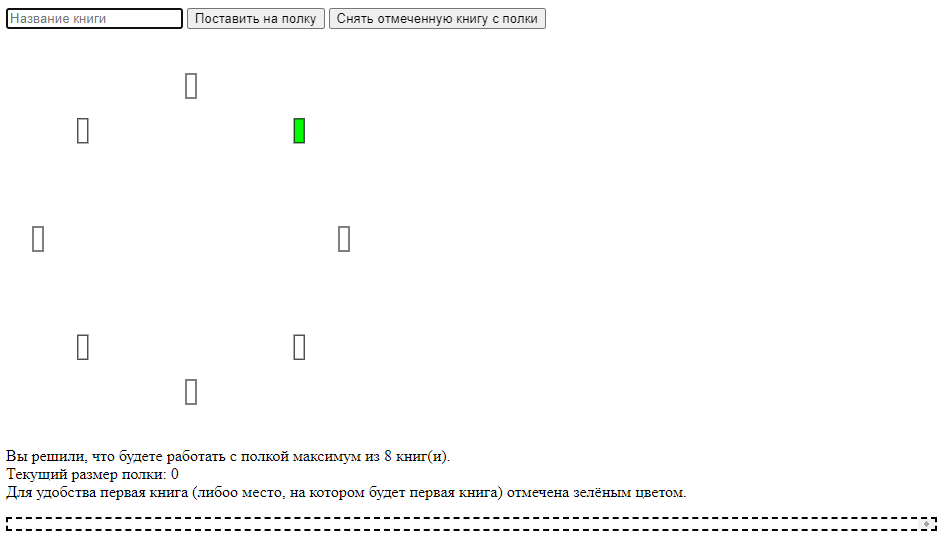
Любые размеры подстраиваются под круг (но не все влезают на страницу):

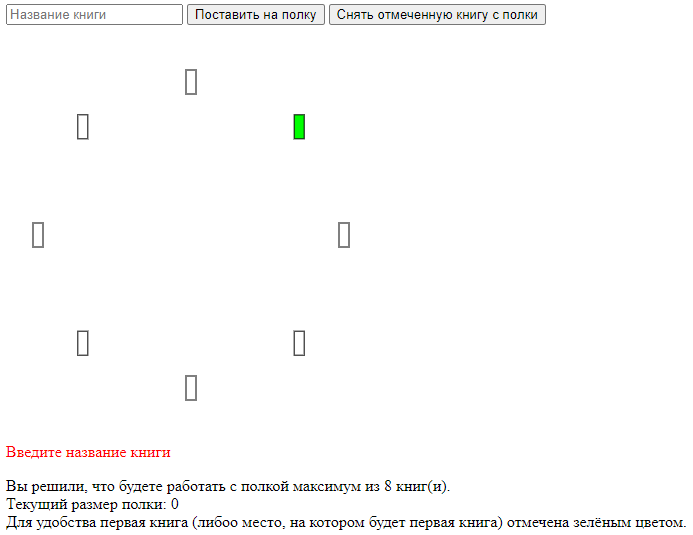


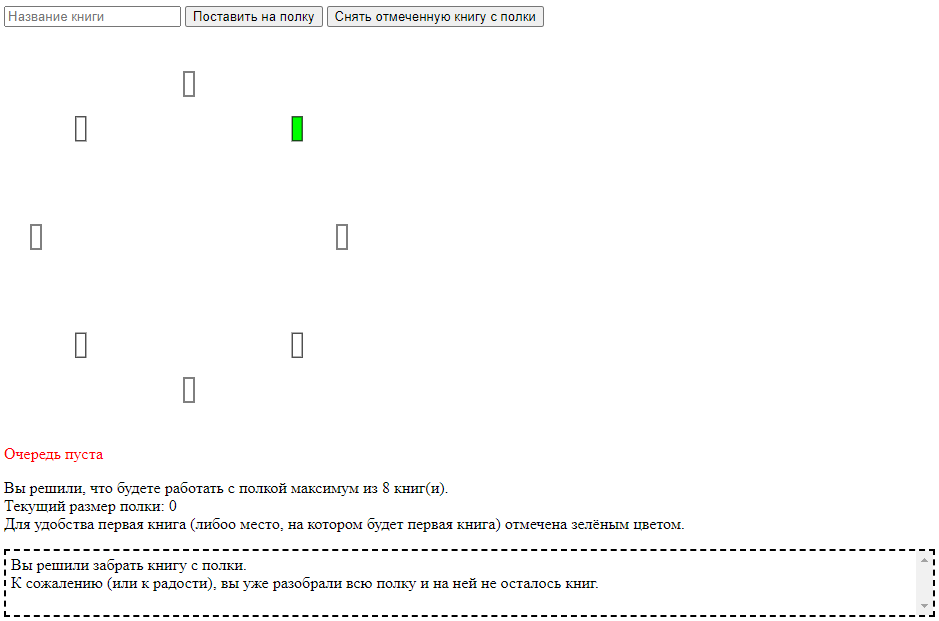


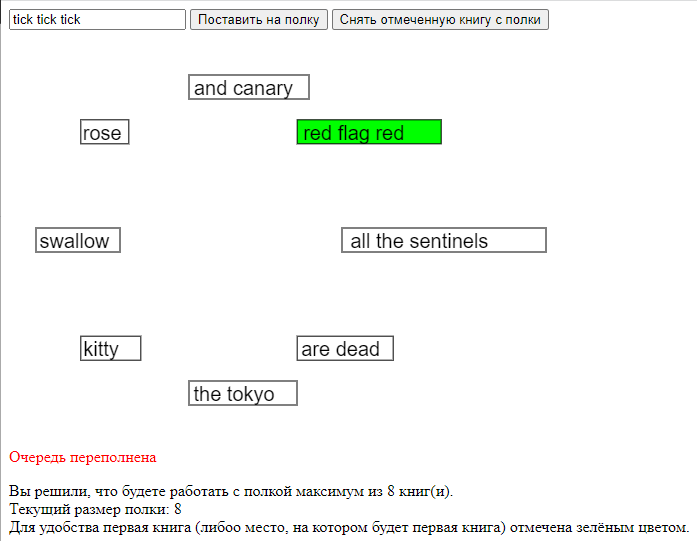
Размеры > 18 уже сложно уместить (слева 20, справа 50):



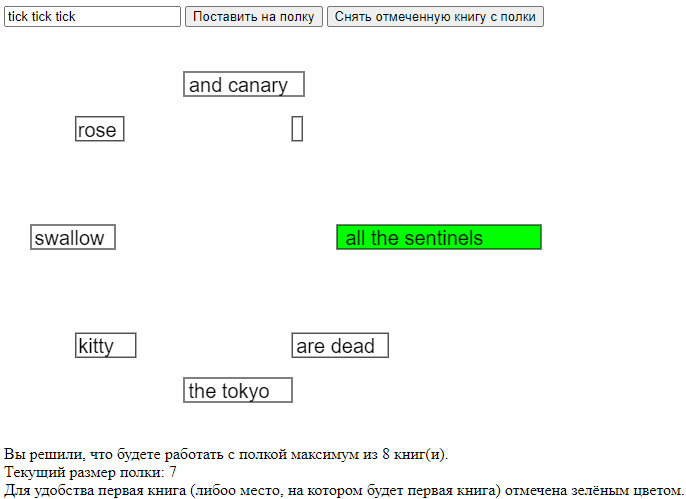


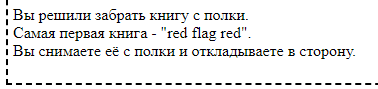


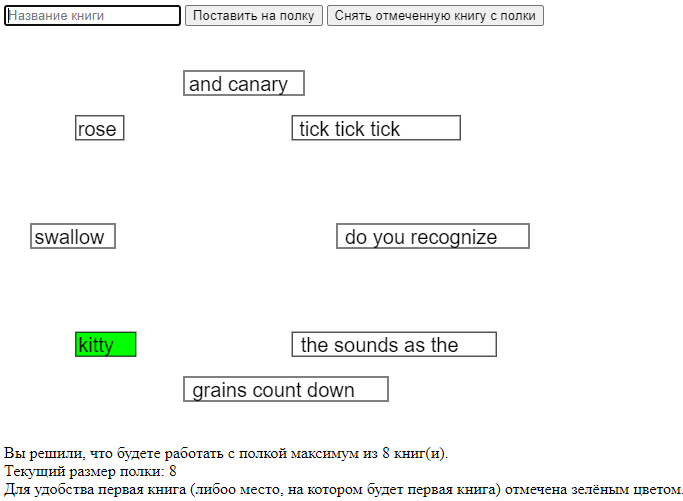


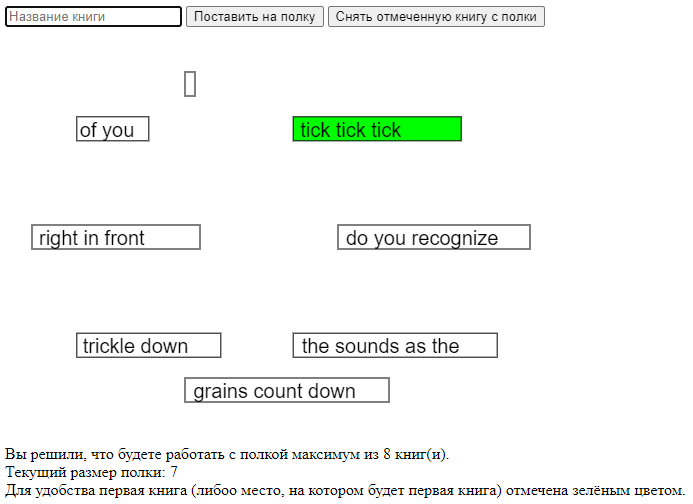


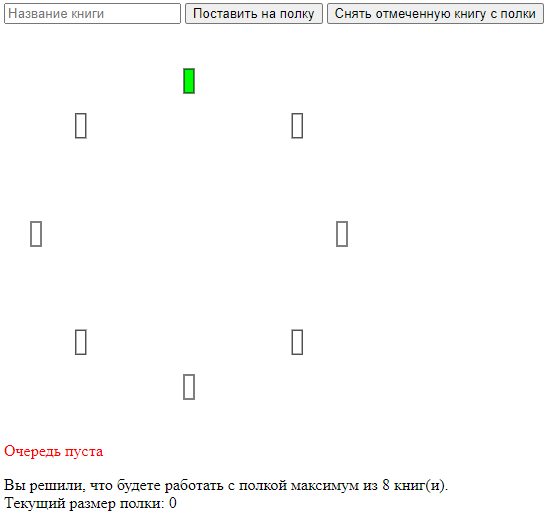


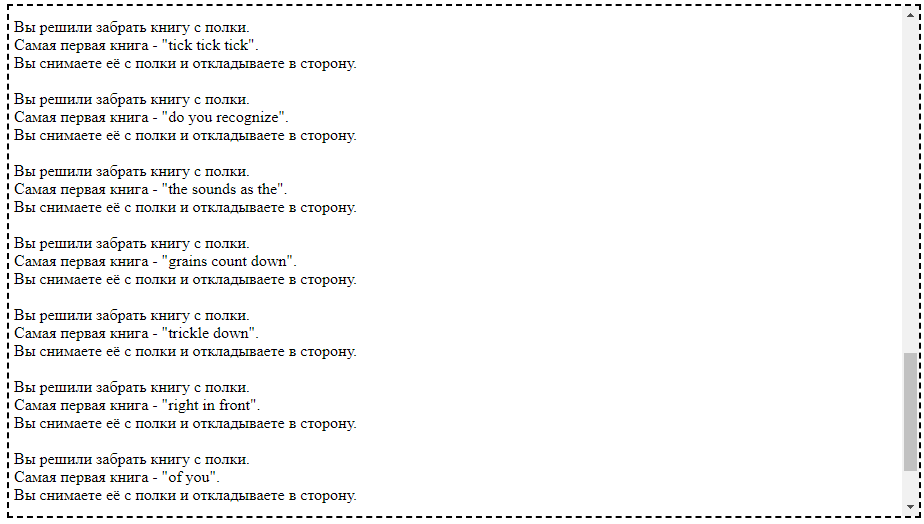




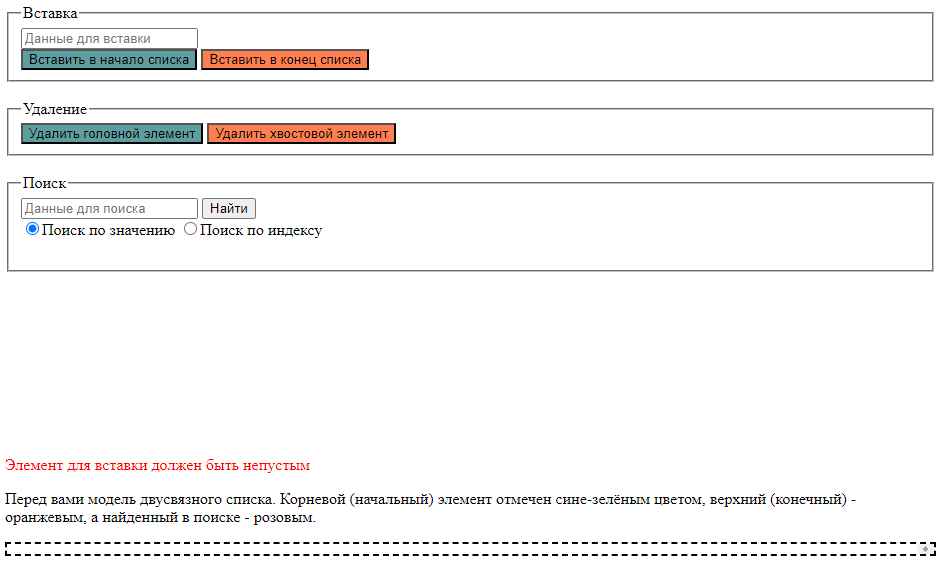


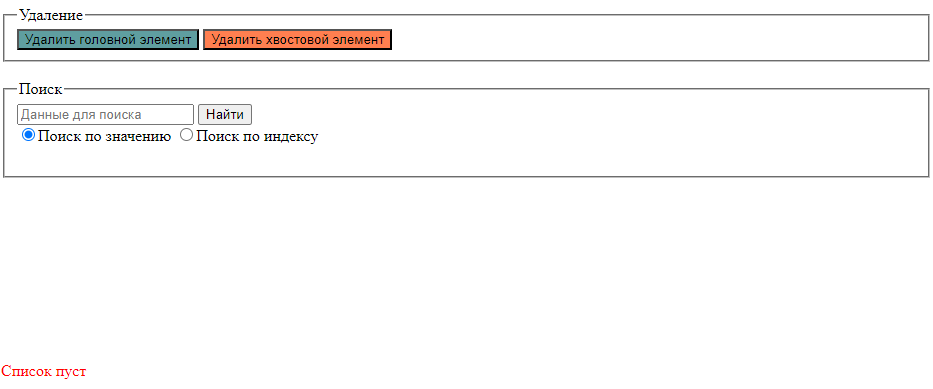


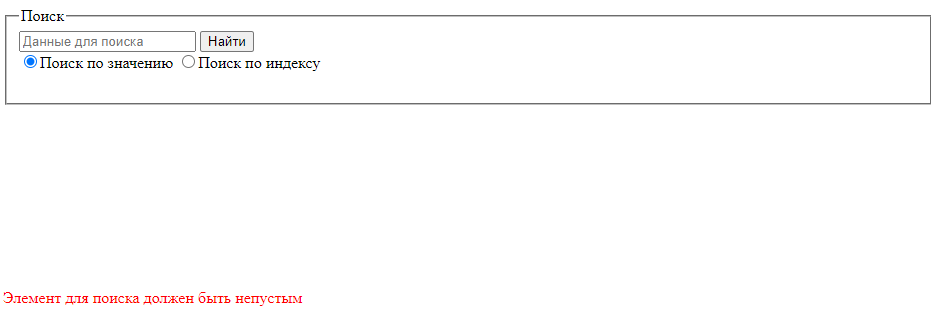


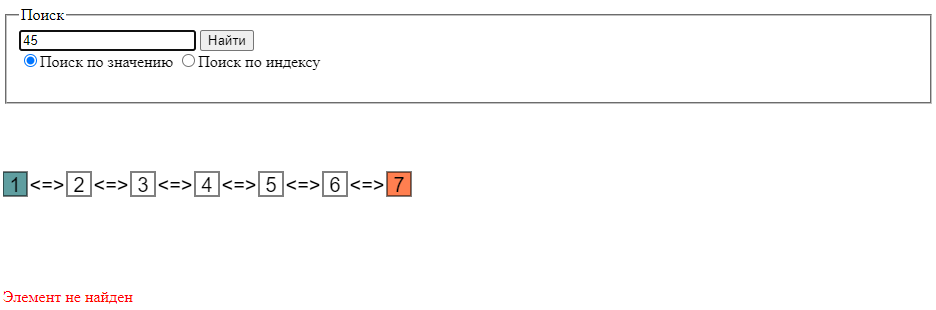


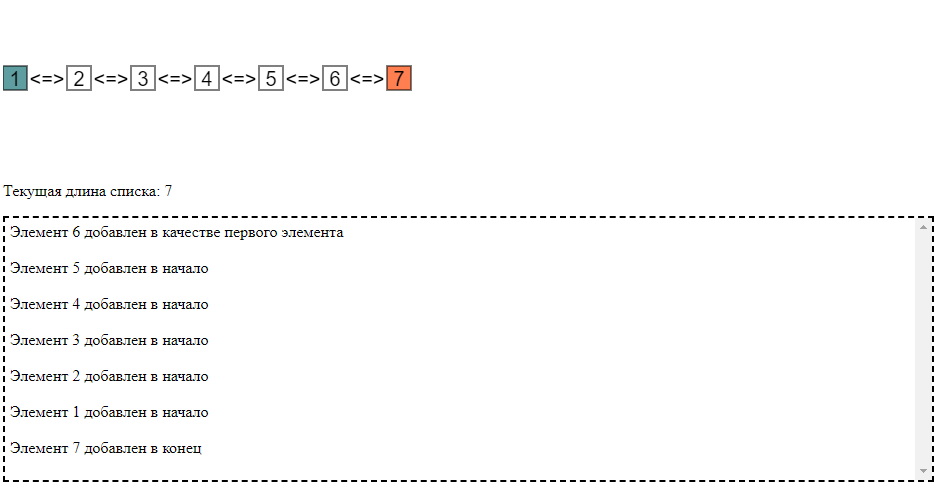
**03.html**





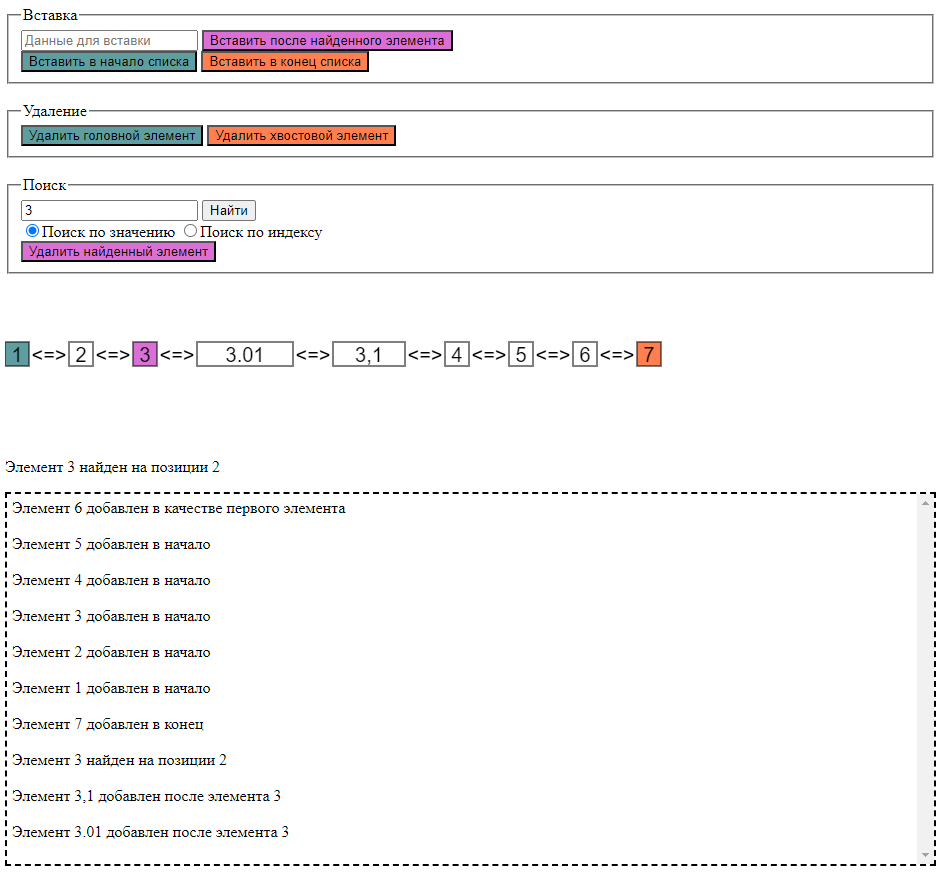


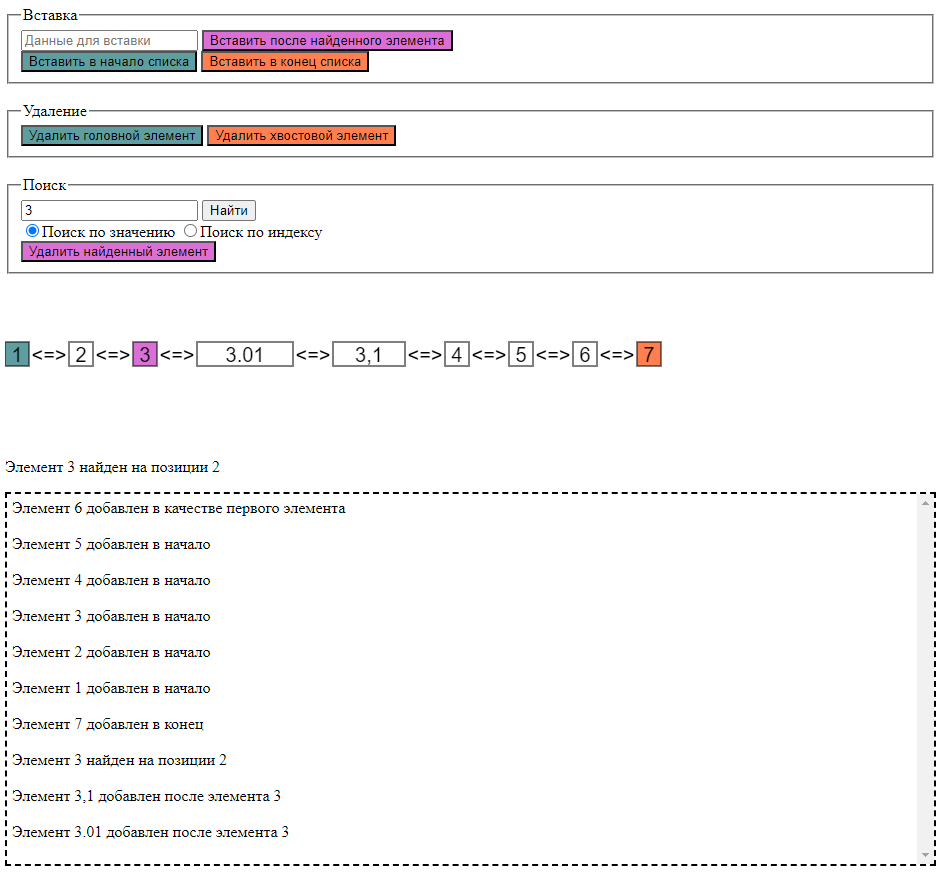






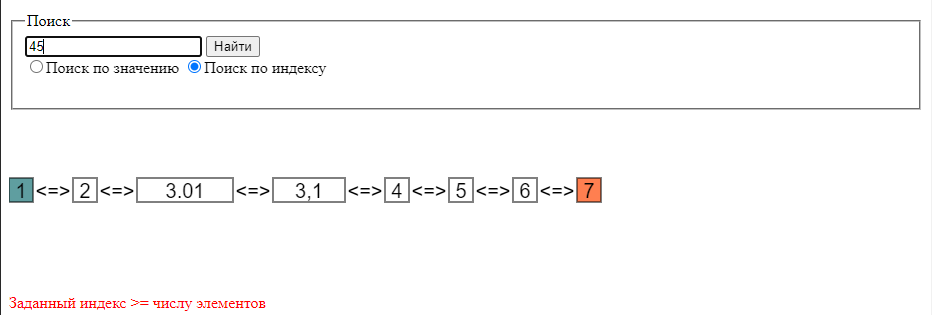


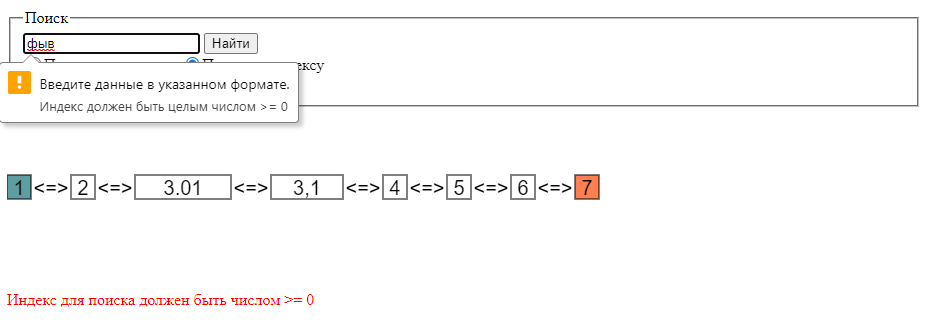


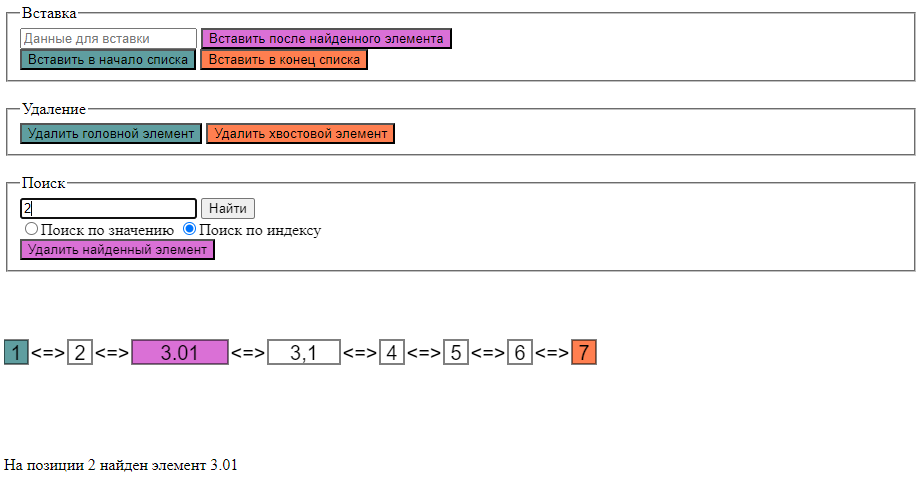












Найденный элемент может быть начальным или конечным:

