**3 ЛАБОРАТОРНАЯ РАБОТА №3**

**«Исследование объектной модели документа (DOM) и системы**

**событий JavaScript»**

**3.1 Цель работы**

Исследовать структуру модели документа DOM. Изучить динамическую объектную модель документа, предоставляемую стандартом DOM и систему событий языка JavaScript, возможность хранения данных на стороне клиента. Приобрести практические навыки работы с событиями JavaScript, деревом документа, Session Storage и Cookies.

**3.2 Вариант задания**

По варианту необходимо реализовать выпадающее меню по наведению на него, а также реализовать часы с форматом даты по шаблону «ЧЧ Месяц ГГ».

**3.3 Ход выполнения работы**

3.3.1 В начале выполнения лабораторной работы было реализовано интерактивное графическое меню сайта. При наведении мыши меню выпадает, а также меняются картинки рядом с пунктами меню. Код файла, содержащего функции для этого представлен в листинге 3.1.

Листинг 3.1 – Файл вывода меню

const listOptions = window.document.querySelectorAll('.list-option');

listOptions.forEach(li => {

addImages(li, 'heart', 'list-img');

li.addEventListener('mouseover', addCat);

li.addEventListener('mouseout', removeCat);

});

function addImages(place, name, imgClass) {

const prev = document.createElement('img');

prev.src = `img/${name}.png`;

prev.classList.add(imgClass);

prev.style.marginRight = '5px';

const next = document.createElement('img');

next.src = `img/${name}.png`;

next.classList.add(imgClass);

next.style.marginLeft = '5px';

place.prepend(prev);

place.append(next);

}

function removeImages(place, imgClass) {

let images = place.querySelectorAll(imgClass);

images.forEach((item) => {

place.removeChild(item);

});

}

function addCat(event) {

const place = event.currentTarget;

removeImages(place, '.list-img');

if (!place.querySelector('.cat-list-img')) {

addImages(place, 'cat', 'cat-list-img');

}

}

function removeCat(event) {

const place = event.currentTarget;

removeImages(place, '.cat-list-img');

addImages(place, 'heart', 'list-img');

}

if(document.getElementById('submenu-item')){

const submenu = document.getElementById('submenu-item');

submenu.addEventListener('mouseover', showMenu);

submenu.addEventListener('mouseout', hideMenu);

}

function showMenu(event) {

const items = document.querySelectorAll('.submenu');

items.forEach(item => {

item.style.height = "auto";

item.style.overflow = "visible";

item.style.opacity = "1";

});

}

function hideMenu(event) {

const items = document.querySelectorAll('.submenu');

items.forEach(item => {

item.style.height = "0";

item.style.overflow = "hidden";

item.style.opacity = "0";

});

}

На рисунке 3.1 можно увидеть готовый результат выпадающего меню.

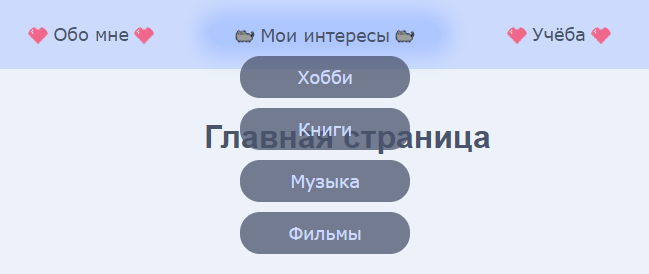


Рисунок 3.1 – Выпадающее меню

3.3.2 Далее были добавлены часы, отображающие помимо времени еще и дату. Код файла, содержащего функции для этого представлен в листинге 3.3.

Листинг 3.2 – Файл вывода часов

const months = [

"января", "февраля", "марта", "апреля", "мая", "июня",

"июля", "августа", "сентября", "октября", "ноября", "декабря"

];

const date = document.createElement("p");

function clockUpdate(){

const curDate = new Date();

const dateEl = window.document.getElementById("clock")

const dateTemp = curDate.getDate().toString()+" "+months[curDate.getMonth()].toString()+" "+curDate.getFullYear().toString();

let minutes = curDate.getMinutes();

if (minutes<10)

minutes = "0" + minutes;

let hours = curDate.getHours();

if (hours<10)

hours = "0" + hours;

let seconds = curDate.getSeconds();

if (seconds<10)

seconds = "0" + seconds;

const timeTmp = hours+":"+minutes+":"+seconds;

date.textContent = dateTemp+" "+timeTmp;

date.style.fontWeight = "bold";

date.style.margin = "0";

date.style.backgroundColor = "#ffa0c5";

date.style.padding = "5px";

date.style.borderRadius = "10px";

dateEl.append(date);

setTimeout("clockUpdate()",1000);

}

window.onload=clockUpdate;

Рисунок 3.2 демонстрирует, как выглядят часы в меню сайта.

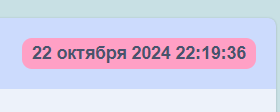


Рисунок 3.2 – Вывод часов

3.3.3 Далее на странице «Контакт» было добавлено поле «Дата рождения», для которого реализован всплывающий снизу элемент «календарь». Код файла, содержащего функции для этого представлен в листинге 3.3.

Листинг 3.3 – Функции проверки формы страницы «Контакт»

let okClicked = false;

const birthdayField = document.getElementById('birthday');

const monthsArray = [

"Январь", "Февраль", "Март", "Апрель", "Май", "Июнь",

"Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"

]

const monthsDays = [

31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

]

const monthSelect = window.document.getElementById("month");

const yearSelect = window.document.getElementById("year");

const birthField = window.document.getElementById("birthday");

birthField.addEventListener("click", showCalendar)

monthSelect.addEventListener("change", outDays)

yearSelect.addEventListener("change", outDays)

const daysContainer = window.document.getElementById("days");

const curDate = new Date();

for (let i = 0; i < monthsArray.length; i++) {

const monthOption = document.createElement("option");

monthOption.textContent = monthsArray[i];

monthOption.value = i;

monthSelect.append(monthOption);

}

for (let i = curDate.getFullYear(); i >= 1950; i--) {

const yearOption = document.createElement("option");

yearOption.textContent = i;

yearOption.value = i;

yearSelect.append(yearOption);

}

function outDays(){

const month = parseInt(monthSelect.value, 10);

const year = parseInt(yearSelect.value, 10);

daysContainer.innerHTML='';

let lastDay;

if((((year%4 === 0)&&(year%100 !== 0))||(year%400 === 0))&&(month===1))

lastDay = monthsDays[month]+1;

else lastDay = monthsDays[month];

for (let i = 1; i <= lastDay; i++) {

const dayBlock = document.createElement("p");

dayBlock.textContent = i;

dayBlock.classList.add("day-block");

dayBlock.addEventListener("click",() => writeDate(i));

daysContainer.append(dayBlock);

}

}

outDays();

const calendar = document.getElementById("calendar-container");

function showCalendar(){

birthdayField.addEventListener("blur", function() {

if (!okClicked) {

birthField.focus();

}

});

calendar.style.display = "block";

calendar.style.opacity = "1";

}

function writeDate(day){

const m = parseInt(monthSelect.value)+1;

const y = parseInt(yearSelect.value);

birthField.value = `${checkDate(day)}.${checkDate(m)}.${y}`;

}

function checkDate(element){

if(element<10) return ("0"+element);

else return element;

}

const button = document.getElementById("calendar-but");

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

okClicked = true;

calendar.style.display = "none";

calendar.style.opacity = "0";

})

Рисунок 3.3 демонстрирует внешний вид выпадающего календаря.

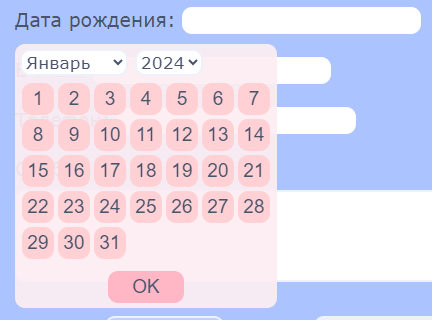


Рисунок 3.3 – Внешний вид выпадающего календаря

3.3.4 Была реализована динамическая проверка корректности заполнения пользователем формы на странице «Контакт». Код файла, содержащего функции для этого представлен в листинге 3.4.

Листинг 3.4 – Функции проверки формы страницы «Контакт»

document.addEventListener('DOMContentLoaded', function() {

const contactForm = document.getElementById('contactForm');

const submitButton = document.getElementById('submit-btn');

submitButton.disabled = true;

if (contactForm) {

contactForm.addEventListener('input', activateSubmit);

}

submitButton.setAttribute('disabled', 'true');

const resetButton = document.getElementById('reset-btn');

resetButton.addEventListener('click', function (event) {

submitButton.setAttribute('disabled', 'true');

resetForm(contactForm);

})

contactForm["fio"].addEventListener('blur', (event) => {

const error = document.getElementById("fio-error");

if (checkFio(contactForm["fio"].value)) {

contactForm["fio"].classList.remove("error");

contactForm["fio"].classList.add("valid");

error.style.display = "none";

} else {

contactForm["fio"].classList.remove("valid");

contactForm["fio"].classList.add("error");

error.style.display = "block";

}

});

contactForm["birthday"].addEventListener('blur', (event) => {

if(contactForm["birthday"].value){

contactForm["birthday"].classList.add("valid");

}

else{

contactForm["birthday"].classList.remove("valid");

}

})

contactForm["email"].addEventListener('blur', (event) => {

const error = document.getElementById("email-error");

if(checkEmail(contactForm["email"].value)){

contactForm["email"].classList.remove("error");

contactForm["email"].classList.add("valid");

error.style.display = "none";

}

else{

contactForm["email"].classList.remove("valid");

contactForm["email"].classList.add("error");

error.style.display = "block";

}

})

contactForm["phone"].addEventListener('blur', (event) => {

const error = document.getElementById("phone-error");

if(checkPhone(contactForm["phone"].value)){

contactForm["phone"].classList.remove("error");

contactForm["phone"].classList.add("valid");

error.style.display = "none";

}

else{

contactForm["phone"].classList.remove("valid");

contactForm["phone"].classList.add("error");

error.style.display = "block";

}

})

contactForm["message"].addEventListener('blur', (event) => {

const error = document.getElementById("message-error");

if(contactForm["message"].value){

contactForm["message"].classList.remove("error");

contactForm["message"].classList.add("valid");

error.style.display = "none";

}

else{

contactForm["message"].classList.remove("valid");

contactForm["message"].classList.add("error");

error.style.display = "block";

}

})

});

function resetForm(contactForm) {

contactForm.reset();

const elements = contactForm.querySelectorAll('.valid, .error');

elements.forEach(element => {

element.classList.remove('valid', 'error');

});

const errorMessages = contactForm.querySelectorAll('.error-message');

errorMessages.forEach(error => {

error.style.display = 'none';

});

}

function checkEmail(email) {

const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

return emailPattern.test(email);

}

function checkPhone(phone) {

const phonePattern = /^\+7\d{10,11}$/;

return phonePattern.test(phone);

}

function checkInput() {

let form = document.forms["contactForm"];

if (form["fio"].value === "") {

return false;

}

if (!form["gender"].value) {

return false;

}

if (!form["birthday"].value) {

return false;

}

if (form["email"].value === "") {

return false;

}

if (form["phone"].value === "") {

return false;

}

if (form["message"].value === "") {

return false;

}

return true;

}

function checkFio(fio) {

fio = fio.trim();

let cnt = 0;

let startPos = 0;

while (fio.indexOf(" ", startPos) >= 0) {

cnt++;

startPos = fio.indexOf(" ", startPos) + 1;

}

if (startPos < fio.length) {

cnt++;

}

return cnt === 3;

}

function activateSubmit(event) {

const submitButton = document.getElementById('submit-btn');

if (checkInput()) {

submitButton.removeAttribute('disabled');

} else {

submitButton.setAttribute('disabled', 'true');

}

}

Рисунок 3.4 демонстрирует, как ведет себя форма при правильном и неправильном вводе.

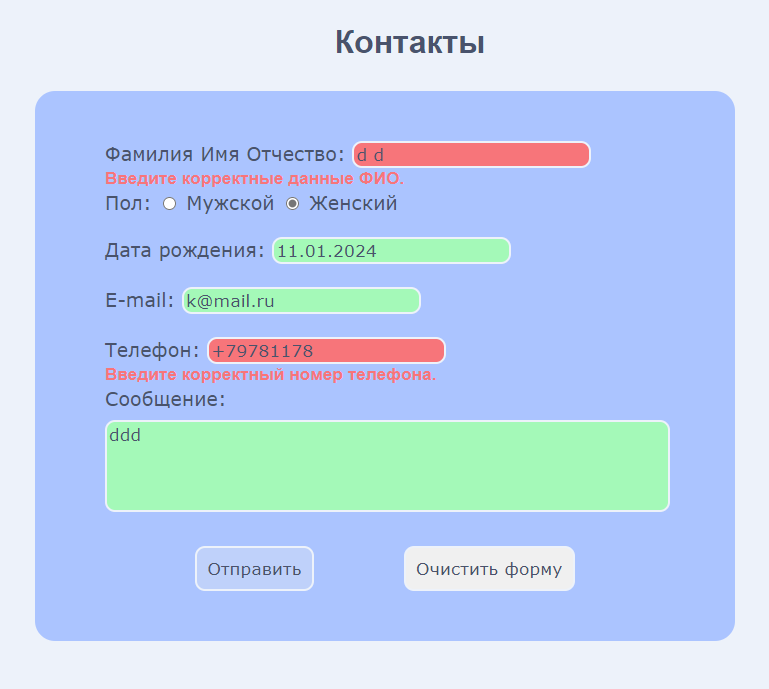


Рисунок 3.4 – Форма «Контакты»

3.3.5 Было реализовано открытие в динамически формируемом новом окне соответствующих больших фото при щелчке мыши по маленьким фото на странице «Фотоальбом». Код файла, содержащего функции для этого представлен в листинге 3.5.

Листинг 3.5 – Функции вывода большого изображения

let formIsClosed = true;

const photos = document.querySelectorAll('img');

photos.forEach(photo => {

photo.addEventListener('click', openPhoto)

})

const bodyMain = document.querySelector('.main-body');

const body = document.querySelector('body');

const form = document.createElement('div')

function openPhoto(event) {

if(formIsClosed){

bodyMain.style.filter = 'blur(5px)';

bodyMain.style.transition = '1s';

console.log(event.target);

addPhotoForm(event.target);

}

}

function addPhotoForm(targetImg){

form.classList.add('photoForm');

form.style.display = 'block';

body.append(form);

form.style.top = `${window.scrollY + 100}px`;

const img = document.createElement('img');

let name = getCharacters(targetImg.src);

img.src = `img/${name}`;

img.style.width = '550px';

img.style.maxHeight = '550px';

img.style.marginTop = '10px';

form.append(img);

body.style.overflow = 'hidden';

const button = document.createElement('button');

button.classList.add('form-but');

button.textContent = 'Закрыть';

button.addEventListener('click', () => closePhotoForm(img, button));

form.append(button);

formIsClosed = false;

}

function closePhotoForm(img, button){

form.remove();

img.remove();

button.remove();

body.style.overflow = 'auto';

bodyMain.style.filter = 'none';

formIsClosed = true;

}

function getCharacters(str) {

const position = str.lastIndexOf("/");

if (position === -1) return '';

return str.substring(position + 1);

}

Рисунок 3.5 демонстрирует, как выполняется увеличение фотографии.

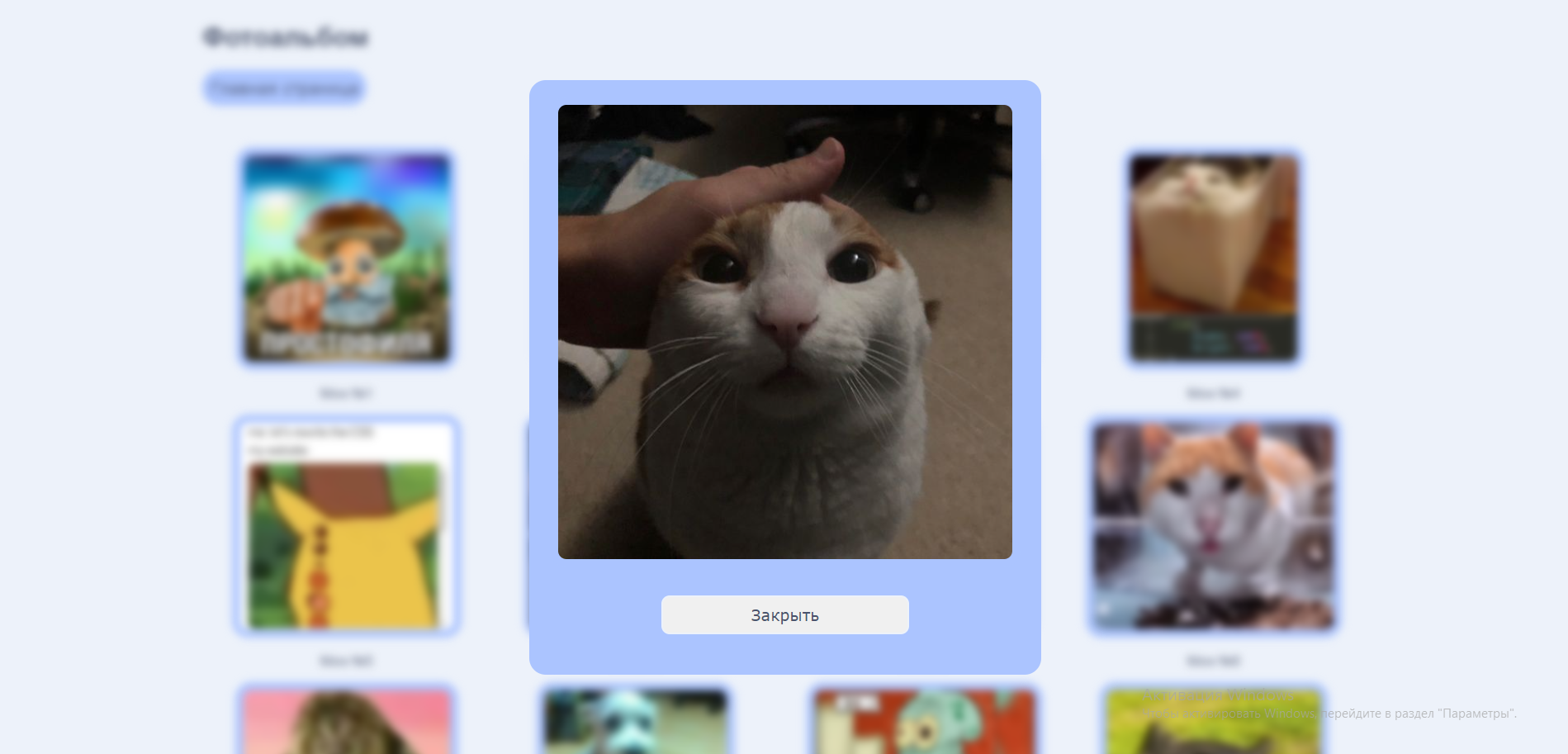


Рисунок 3.5 – Увеличение изображений

3.3.6 Был реализован вывод истории посещения страницы в конкретной сессии и за все время использования сайта. Код файла, содержащего функции для этого представлен в листинге 3.6.

Листинг 3.6 – Функции по выводу истории посещений

const pagesArrAll = {

"main-views-all": "Главная страница",

"aboutme-views-all": "Обо мне",

"interests-views-all": "Мои интересы",

"album-views-all": "Фотоальбом",

"contact-views-all": "Контакт",

"study-views-all": "Учёба",

"test-views-all": "Тест",

"history-views-all": "История просмотра"

};

const pagesArrCur = {

"main-views-cur": "Главная страница",

"aboutme-views-cur": "Обо мне",

"interests-views-cur": "Мои интересы",

"album-views-cur": "Фотоальбом",

"contact-views-cur": "Контакт",

"study-views-cur": "Учёба",

"test-views-cur": "Тест",

"history-views-cur": "История просмотра"

};

let storage = window.sessionStorage;

function increaseCounter() {

let cntCur = storage.getItem(document.title);

cntCur++;

storage.setItem(document.title, cntCur);

let cntAll = getCookie(document.title)+1;

setCookie(document.title, cntAll, 5);

}

function updateCounter() {

const tdsCur = document.querySelectorAll('.td-cur');

tdsCur.forEach(td => {

let cnt = storage.getItem(pagesArrCur[td.id]);

if(cnt === null){

storage.setItem(pagesArrCur[td.id], "0");

}

td.textContent = storage.getItem(pagesArrCur[td.id]);

});

const tdsAll = document.querySelectorAll('.td-all');

tdsAll.forEach(td => {

console.log(pagesArrAll[td.id]);

let cnt = getCookie(pagesArrAll[td.id]);

if(!cnt){

setCookie(pagesArrAll[td.id], 0, 5);

}

td.textContent = getCookie(pagesArrAll[td.id]).toString();

})

}

function setCookie(name, value, days) {

document.cookie = name + '=' + value + '; max-age=' + (days \* 24 \* 60 \* 60) + ';';

}

function getCookie(name){

let cookieStr = document.cookie;

let key = cookieStr.search(name+"=");

let index = key+parseInt(name.length)+1;

let value="";

for(let i = index; i <= cookieStr.length; i++){

if(cookieStr[i]!==';') value = value+cookieStr[i];

}

return parseInt(value);

}

Рисунок 3.5 демонстрирует, как отображается вывод истории посещений.

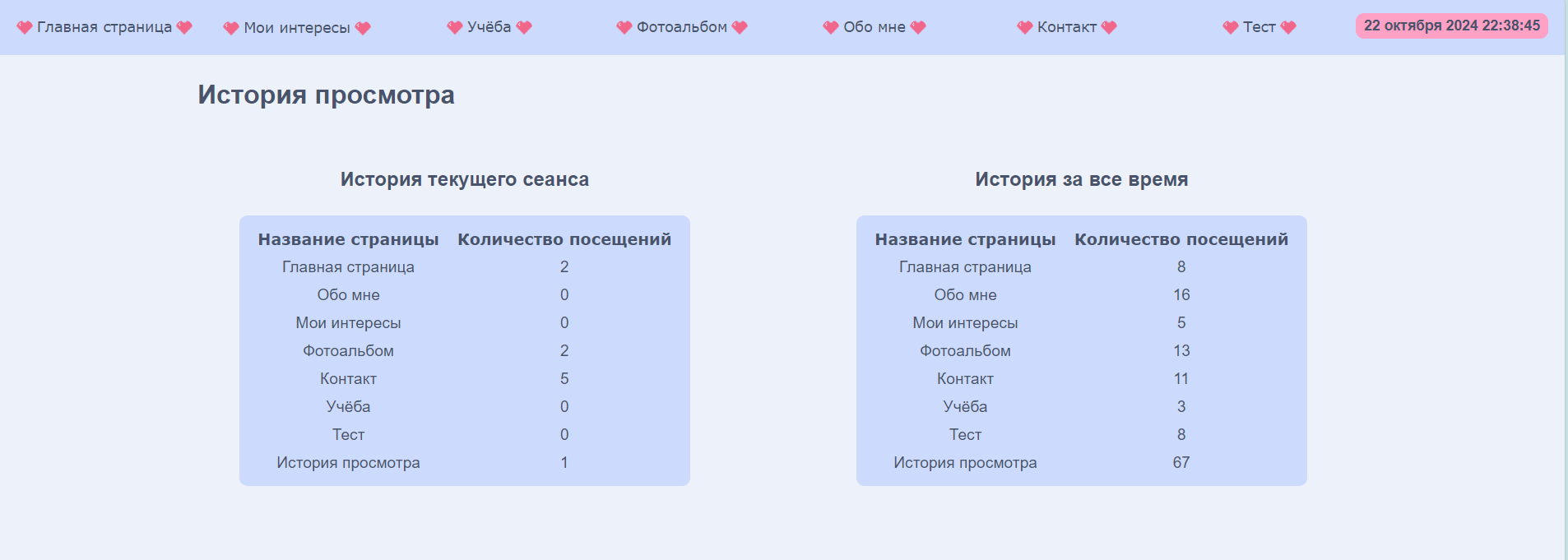


Рисунок 3.5 – Вывод истории посещений

**Выводы**

В ходе лабораторной работы была исследована структура модели документа DOM. Также была изучена динамическая объектная модель документа, предоставляемая стандартом DOM и системе событий языка JavaScript, возможность хранения данных на стороне клиента. Были приобретены практические навыки работы с событиями JavaScript, деревом документа, Session Storage и Cookies.