**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.ММ.ГГГГ День недели. Реализовать динамическую проверку полей ввода на странице контакт, а также добавить поле дата рождения. Добавить историю посещений с использованием session storage и cookies.

**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

document.addEventListener('DOMContentLoaded', () => {

document.querySelectorAll('.main-menu .has-dropdown').forEach(item => {

item.addEventListener('mouseover', () => {

item.querySelector('.dropdown').style.display = 'block';

});

item.addEventListener('mouseout', () => {

item.querySelector('.dropdown').style.display = 'none';

});

});

});

const *menuItems* = *document*.querySelectorAll('.main-menu li');  
  
*menuItems*.forEach(item => {  
 const link = item.querySelector('a');  
 const img = link.querySelector('img');  
  
 item.addEventListener('mouseover', () => {  
 if (!item.classList.contains('selected')) {  
 img.src = "images/menu/menuout.png";  
 }  
 });  
  
 item.addEventListener('mouseout', () => {  
 if (!item.classList.contains('selected')) {  
 img.src = "images/menu/menuover.png";  
 }  
 });  
});

На рисунке 3.1 можно увидеть готовый результат выпадающего меню.

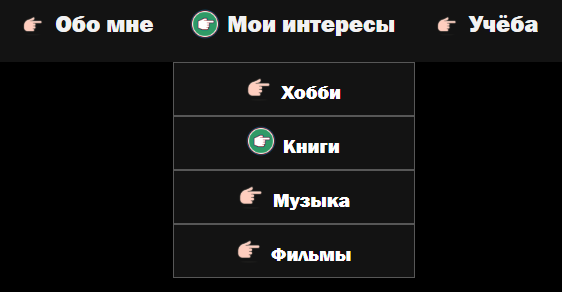


Рисунок 3.1 – Выпадающее меню

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

function updateDateTime() {  
 const now = new *Date*();  
 const days = ['Воскресенье', 'Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота'];  
 const formattedDateTime =  
 ('0' + now.getHours()).slice(-2) + ':' +  
 ('0' + now.getMinutes()).slice(-2) + ':' +  
 ('0' + now.getSeconds()).slice(-2) + ' ' +  
 ('0' + now.getDate()).slice(-2) + '.' +  
 ('0' + (now.getMonth() + 1)).slice(-2) + '.' +  
 now.getFullYear() + ' ' +  
 days[now.getDay()];  
 *document*.getElementById('datetime').textContent = formattedDateTime;  
}  
setInterval(updateDateTime, 1000);  
updateDateTime();

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»

function addBirthdateField() {

const birthdateDiv = document.getElementById('birthdate');

const headerContainer = document.createElement('div');

headerContainer.className = 'header-container';

const monthSelect = document.createElement('select');

monthSelect.name = 'month';

monthSelect.required = true;

monthSelect.innerHTML = `

<option value="01">Январь</option>

<option value="02">Февраль</option>

<option value="03">Март</option>

<option value="04">Апрель</option>

<option value="05">Май</option>

<option value="06">Июнь</option>

<option value="07">Июль</option>

<option value="08">Август</option>

<option value="09">Сентябрь</option>

<option value="10">Октябрь</option>

<option value="11">Ноябрь</option>

<option value="12">Декабрь</option>

`;

headerContainer.appendChild(monthSelect);

const yearSelect = document.createElement('select');

yearSelect.name = 'year';

yearSelect.required = true;

yearSelect.innerHTML = '<option value="" selected>Год</option>';

const currentYear = new Date().getFullYear();

for (let i = currentYear; i >= currentYear - 100; i--) {

yearSelect.innerHTML += `<option value="${i}">${i}</option>`;

}

headerContainer.appendChild(yearSelect);

birthdateDiv.appendChild(headerContainer);

const weekdayContainer = document.createElement('div');

weekdayContainer.className = 'weekday-container';

const weekdays = ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Вс'];

weekdays.forEach(weekday => {

const dayElement = document.createElement('div');

dayElement.className = 'weekday';

dayElement.innerText = weekday;

weekdayContainer.appendChild(dayElement);

});

birthdateDiv.appendChild(weekdayContainer);

const dayContainer = document.createElement('div');

dayContainer.className = 'day-container';

dayContainer.id = 'day-container';

birthdateDiv.appendChild(dayContainer);

const buttonContainer = document.createElement('div');

buttonContainer.className = 'button-container';

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

closeButton.innerText = 'Закрыть';

closeButton.onclick = toggleCalendar;

buttonContainer.appendChild(closeButton);

birthdateDiv.appendChild(buttonContainer);

setCurrentDate();

monthSelect.addEventListener('change', updateDays);

yearSelect.addEventListener('change', updateDays);

}

function setCurrentDate() {

const now = new Date();

const currentMonth = String(now.getMonth() + 1).padStart(2, '0');

const currentYear = String(now.getFullYear());

const currentDay = String(now.getDate()).padStart(2, '0');

document.querySelector('select[name="month"]').value = currentMonth;

document.querySelector('select[name="year"]').value = currentYear;

updateDays();

const dayElement = Array.from(document.querySelectorAll('.day')).find(day => day.innerText === currentDay);

if (dayElement) {

dayElement.classList.add('selected');

}

}

function toggleCalendar() {

const calendar = document.getElementById('birthdate');

if (calendar.style.display === 'block') {

calendar.style.display = 'none';

} else {

calendar.style.display = 'block';

const birthdateInput = document.getElementById('birthdate-input');

calendar.style.top = `${birthdateInput.offsetTop + birthdateInput.offsetHeight}px`;

calendar.style.left = `${birthdateInput.offsetLeft}px`;

}

}

function confirmDate() {

const month = document.querySelector('select[name="month"]').value;

const day = document.querySelector('.day.selected')?.innerText;

const year = document.querySelector('select[name="year"]').value;

if (month && day && year) {

document.getElementById('birthdate-input').value = `${month}/${day}/${year}`;

toggleCalendar();

} else {

alert("Пожалуйста, выберите полную дату.");

}

}

function selectDay(dayElement) {

const previouslySelected = document.querySelector('.day.selected');

if (previouslySelected) {

previouslySelected.classList.remove('selected');

}

dayElement.classList.add('selected');

confirmDate();

}

function updateDays() {

const month = document.querySelector('select[name="month"]').value;

const year = document.querySelector('select[name="year"]').value;

const dayContainer = document.getElementById('day-container');

if (!month || !year) {

dayContainer.innerHTML = '';

return;

}

const daysInMonth = new Date(year, month, 0).getDate();

const firstDayOfMonth = new Date(year, month - 1, 1).getDay();

const adjustedFirstDay = (firstDayOfMonth + 6) % 7; // Adjust for the week starting on Monday

dayContainer.innerHTML = '';

for (let i = 0; i < adjustedFirstDay; i++) {

const emptyDiv = document.createElement('div');

emptyDiv.className = 'day empty';

dayContainer.appendChild(emptyDiv);

}

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

const dayDiv = document.createElement('div');

dayDiv.className = 'day';

dayDiv.innerText = String(i).padStart(2, '0');

dayDiv.onclick = () => selectDay(dayDiv);

dayContainer.appendChild(dayDiv);

}

const totalSlots = adjustedFirstDay + daysInMonth;

const remainingSlots = totalSlots % 7;

if (remainingSlots > 0) {

for (let i = remainingSlots; i < 7; i++) {

const emptyDiv = document.createElement('div');

emptyDiv.className = 'day empty';

dayContainer.appendChild(emptyDiv);

}

}

}

Рисунок 3.3 демонстрирует внешний вид выпадающего календаря.

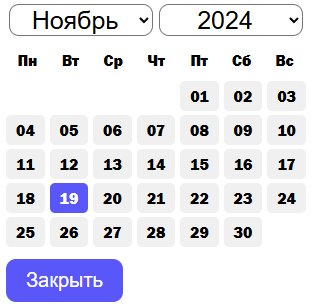


Рисунок 3.3 – Внешний вид выпадающего календаря

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»

isFioCorrect = false;  
isBirthdateCorrect = false;  
isAgeCorrect = false;  
isEmailCorrect = false;  
isPhoneCorrect = false;  
  
*window*.onload = function () {  
 const form = *document*.querySelector('form');  
 const fioInput = *document*.querySelector("input[name='fio']");  
 const birthdateInput = *document*.querySelector("#birthdate-input");  
 const ageSelect = *document*.querySelector("select[name='age']");  
 const emailInput = *document*.querySelector("input[name='email']");  
 const phoneInput = *document*.querySelector("input[name='phone']");  
 const submitButton = *document*.querySelector("button[type='submit']");  
 const confirmDay = *document*.querySelector("div[class='day']");  
  
 const validateFio = () => {  
 const fioRegex = new *RegExp*("[A-Za-zА-Яа-я]{2,} [A-Za-zА-Яа-я]{2,} [A-Za-zА-Яа-я]{2,}");  
 if (!fioRegex.test(fioInput.value.trim())) {  
 showError(fioInput, "Не так тебя зовут");  
 isFioCorrect = false;  
 return false;  
 }  
 showValid(fioInput);  
 isFioCorrect = true;  
 return true;  
 };  
  
 const validateNumber = () => {  
 const phoneRegex = new *RegExp*("[\\+][37][0-9]{8,10}");  
 if (!phoneRegex.test(phoneInput.value.trim())) {  
 showError(phoneInput, "Неправильно набран номер");  
 isPhoneCorrect = false;  
 return false;  
 }  
 showValid(phoneInput);  
 isPhoneCorrect = true;  
 return true;  
 };  
  
 const validateEmail = () => {  
 const emailRegex = new *RegExp*("[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}");  
 if (!emailRegex.test(emailInput.value.trim())) {  
 showError(emailInput, "Введите корректный email");  
 isEmailCorrect = false;  
 return false;  
 }  
 showValid(emailInput);  
 isEmailCorrect = true;  
 return true;  
 };  
  
 const validateBirthdate = () => {  
 if (birthdateInput.value.trim() === '') {  
 showError(birthdateInput, "Выберите дату рождения");  
 isBirthdateCorrect = false;  
 return false;  
 }  
 showValid(birthdateInput);  
 isBirthdateCorrect = true;  
 return true;  
 };  
  
 const validateAge = () => {  
 if (ageSelect.value === '' || ageSelect.value === 'Не выбран') {  
 showError(ageSelect, "Выберите возраст");  
 isAgeCorrect = false;  
 return false;  
 }  
 showValid(ageSelect);  
 isAgeCorrect = true;  
 return true;  
 };  
  
 const validateForm = () => {  
 submitButton.disabled = !(isFioCorrect && isBirthdateCorrect && isAgeCorrect && isEmailCorrect && isPhoneCorrect);  
 };  
  
 const showError = (input, message) => {  
 input.classList.add('invalid');  
 input.classList.remove('valid');  
 let errorSpan = input.nextElementSibling;  
 errorSpan.textContent = message;  
 errorSpan.style.display = 'block';  
 };  
  
 const hideError = (input) => {  
 input.classList.remove('invalid');  
 input.classList.remove('valid');  
 const errorSpan = input.nextElementSibling;  
 if (errorSpan && errorSpan.classList.contains('error-message')) {  
 errorSpan.style.display = 'none';  
 }  
 };  
  
 const showValid = (input) => {  
 input.classList.remove('invalid');  
 input.classList.add('valid');  
 const errorSpan = input.nextElementSibling;  
 if (errorSpan && errorSpan.classList.contains('error-message')) {  
 errorSpan.style.display = 'none';  
 }  
 };  
  
 fioInput.addEventListener('input', () => { validateFio(); validateForm(); });  
 fioInput.addEventListener('blur', () => { validateFio(); validateForm(); });  
 birthdateInput.addEventListener('change', () => { validateBirthdate(); validateForm(); });  
 birthdateInput.addEventListener('input', () => { validateBirthdate(); validateForm(); });  
 birthdateInput.addEventListener('blur', () => { validateBirthdate(); validateForm(); });  
 confirmDay.addEventListener('click', () => { validateBirthdate(); validateForm();});  
 ageSelect.addEventListener('change', () => { validateAge(); validateForm(); });  
 ageSelect.addEventListener('blur', () => { validateAge(); validateForm(); });  
 emailInput.addEventListener('input', () => { validateEmail(); validateForm(); });  
 emailInput.addEventListener('blur', () => { validateEmail(); validateForm(); });  
 phoneInput.addEventListener('input', () => { validateNumber(); validateForm(); });  
 phoneInput.addEventListener('blur', () => { validateNumber(); validateForm(); });  
  
 form.addEventListener('input', validateForm);  
};

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

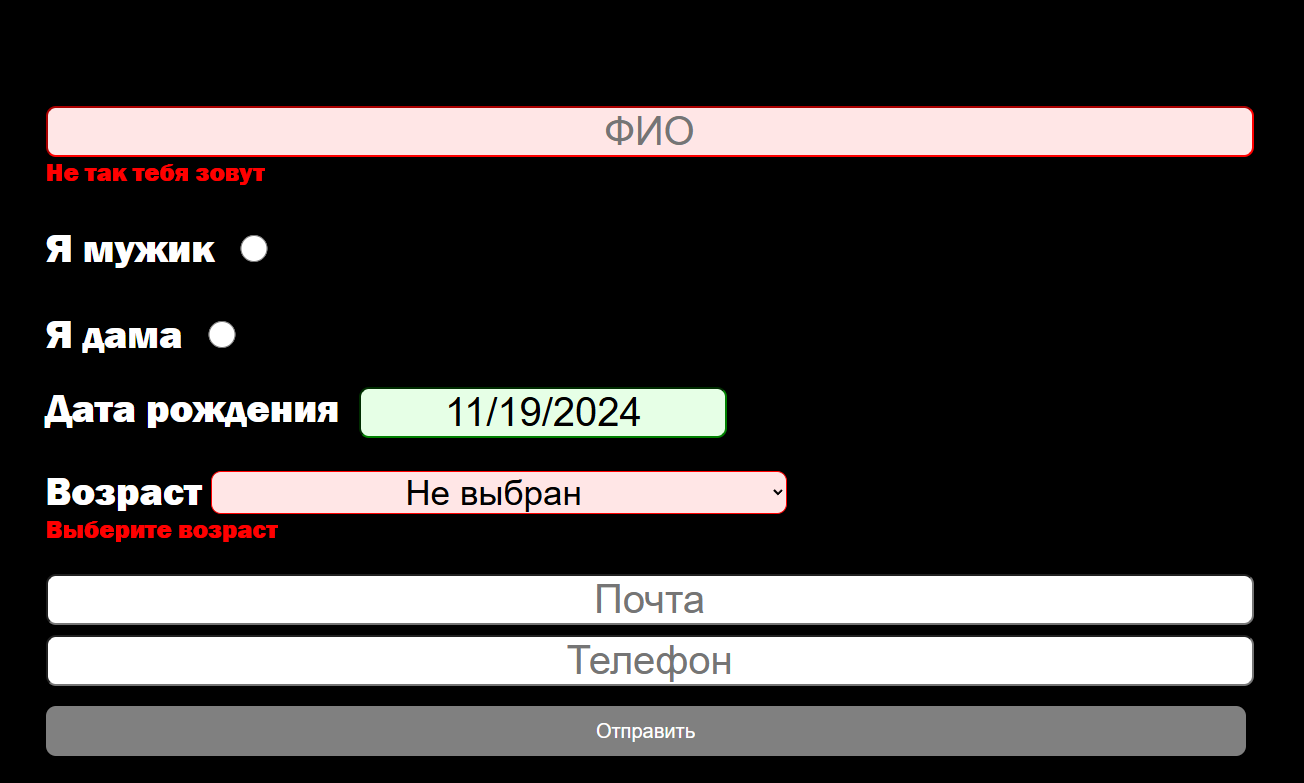


Рисунок 3.4 – Форма «Контакты»

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

function showImage(src) {  
 var largePhotoDiv = *document*.createElement('div');  
 largePhotoDiv.className = 'large-photo-container';  
 largePhotoDiv.innerHTML = '<img src="' + src + '" alt="Большое фото">';  
  
 largePhotoDiv.addEventListener('click', function(event) {  
 if (event.target === largePhotoDiv) {  
 *document*.body.removeChild(largePhotoDiv);  
 }  
 });  
  
 *document*.body.appendChild(largePhotoDiv);  
}  
  
function showImages() {  
 var images = [];  
 var titles = [];  
  
 for (var i = 1; i < 16; i++) {  
 images[i] = `images/album/a (${i}).JPEG`;  
 titles[i] = `Фото ${i}`;  
 }  
  
 var album = *document*.getElementById('photo-album');  
 for (var i = 0; i < 3; i++) {  
 var container = *document*.createElement('div');  
 container.className = 'container';  
 for (var j = 1; j <= 5; j++) {  
 var card = *document*.createElement('div');  
 card.className = 'card';  
  
 var img = *document*.createElement('img');  
 img.className = 'album-image';  
 img.src = images[i\*5+j];  
 img.alt = '';  
 img.title = titles[i\*5+j];  
 img.onclick = (function(src) {  
 return function() {  
 showImage(src);  
 };  
 })(images[i\*5+j]);  
  
 var textContainer = *document*.createElement('div');  
 var title = *document*.createElement('h1');  
 title.className = 'album-image-text';  
 title.textContent = titles[i\*5+j];  
  
 textContainer.appendChild(title);  
 card.appendChild(img);  
 card.appendChild(textContainer);  
 container.appendChild(card);  
 }  
 album.appendChild(container);  
 }  
}  
  
*document*.addEventListener('DOMContentLoaded', function() {  
 showImages();  
});

Рисунок 3.5 демонстрирует, как выполняется увеличение фотографии.

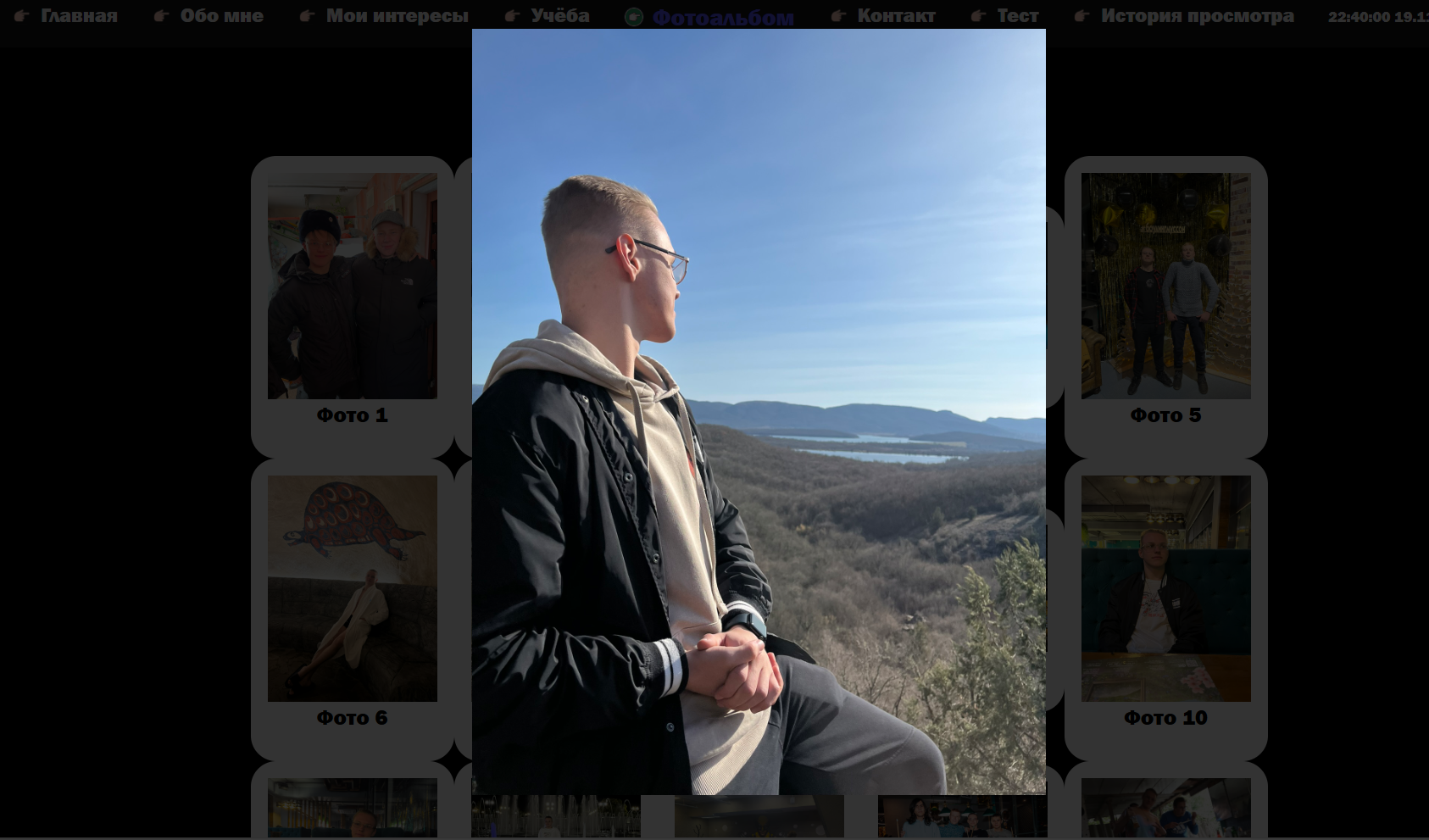


Рисунок 3.5 – Увеличение изображений

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

function updateSessionHistory(page) {  
 const sessionHistory = *JSON*.parse(*sessionStorage*.getItem('sessionHistory')) || {};  
 sessionHistory[page] = (sessionHistory[page] || 0) + 1;  
 *sessionStorage*.setItem('sessionHistory', *JSON*.stringify(sessionHistory));  
 *console*.log("Session History Updated:", sessionHistory);  
}  
  
function updateAllTimeHistory(page) {  
 const allTimeHistory = *JSON*.parse(getCookie('allTimeHistory')) || {};  
 allTimeHistory[page] = (allTimeHistory[page] || 0) + 1;  
 setCookie('allTimeHistory', *JSON*.stringify(allTimeHistory), 365);  
 *console*.log("All Time History Updated:", allTimeHistory);  
}  
  
function updateHistory(page) {  
 updateSessionHistory(page);  
 updateAllTimeHistory(page);  
}  
  
*document*.addEventListener('DOMContentLoaded', () => {  
 const page = *window*.location.pathname;  
 updateHistory(page);  
});

function setCookie(name, value, expirationDays) {  
 const date = new *Date*();  
 date.setTime(date.getTime() + (expirationDays \* 24 \* 60 \* 60 \* 1000));  
 const expires = "expires=" + date.toUTCString();  
 *document*.cookie = `${name}=${value};${expires};path=/`;  
}  
  
function getCookie(name) {  
 const nameEQ = name + "=";  
 const ca = *document*.cookie.split(';');  
 for (let i = 0; i < ca.length; i++) {  
 let c = ca[i];  
 while (c.charAt(0) === ' ') c = c.substring(1, c.length);  
 if (c.indexOf(nameEQ) === 0) {  
 return c.substring(nameEQ.length, c.length);  
 }  
 }  
 return null;  
}

*document*.addEventListener('DOMContentLoaded', () => {  
 const sessionHistory = *JSON*.parse(*sessionStorage*.getItem('sessionHistory')) || {};  
 const allTimeHistory = *JSON*.parse(getCookie('allTimeHistory')) || {};  
  
 const sessionTbody = *document*.querySelector('#sessionHistory tbody');  
 sessionTbody.innerHTML = '';  
 for (const page in sessionHistory) {  
 const tr = *document*.createElement('tr');  
 tr.innerHTML = `<td>${page}</td><td>${sessionHistory[page]}</td>`;  
 sessionTbody.appendChild(tr);  
 }  
  
 const allTimeTbody = *document*.querySelector('#allTimeHistory tbody');  
 allTimeTbody.innerHTML = '';  
 for (const page in allTimeHistory) {  
 const tr = *document*.createElement('tr');  
 tr.innerHTML = `<td>${page}</td><td>${allTimeHistory[page]}</td>`;  
 allTimeTbody.appendChild(tr);  
 }  
});

Рисунок 3.5 демонстрирует, как отображается вывод истории посещений.

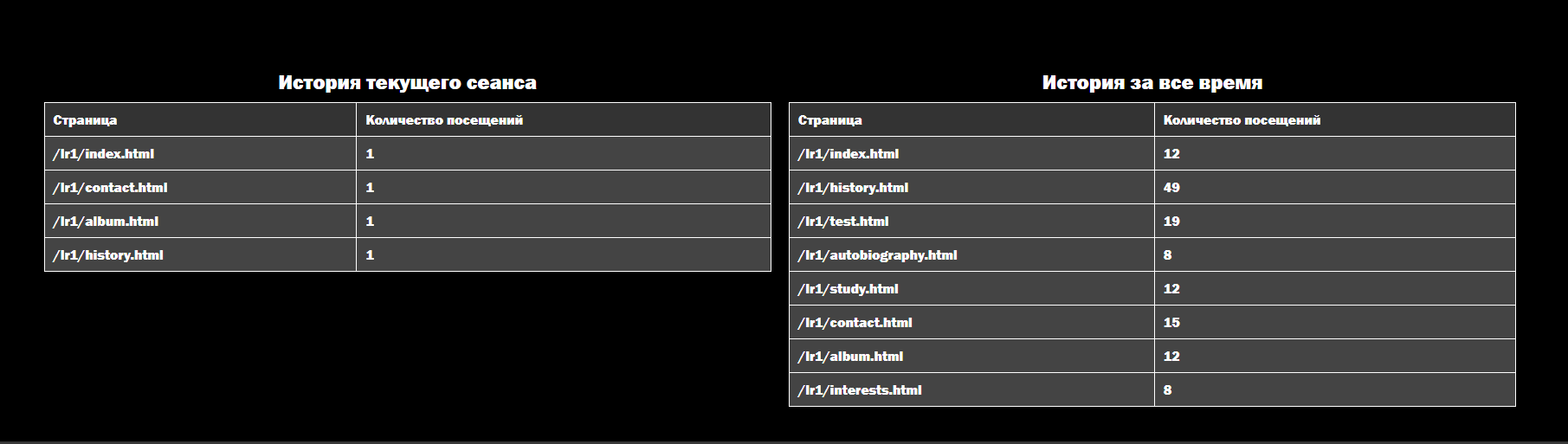


Рисунок 3.5 – Вывод истории посещений

**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.