МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

ОТЧЕТ
о выполнении лабораторной работы №3
«Исследование объектной модели документа
(DOM) и системы событий JavaScript»
по дисциплине
«Веб-технологии»

Выполнил:
ст.гр. ИС/б-23-1-о
Баймурадов Т. Р.

Севастополь, 2025

1. Цель

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, Local Storage и Cookies.

2. Постановка задания

1. Реализовать интерактивное графическое меню сайта, в соответствии со следующими требованиями:

- при наведении мыши на соответствующий пункт меню изображение, соответствующее этому пункту меняется на другое (в том случае если этот пункт не активен, т.е. не загружена соответствующая ему страница);
- сделать пункт «Мои интересы» в виде выпадающего меню, каждый элемент которого ведет на соответствующую ему подсекцию данной страницы. Выпадающее меню реализовать с помощью тегов и . Данный список должен раскрываться в зависимости от варианта задания:
- 1: Выпадающее меню раскрывается по наведению

2. Реализовать отображение в области меню сайта текущих даты и времени (обновление времени 1 раз в секунду). Формат даты определяется как остаток от деления последней цифры зачетной книжки на 4:

- 3: ЧЧ Месяц ГГГГ

(«День недели» – полная запись дня недели на русском языке, «Месяц» – полная запись названия месяца на русском языке)

3. На странице «Контакт» добавить поле «Дата рождения», для которого реализовать всплывающий снизу элемент «календарь». При этом элемент должен давать возможность менять дату, месяц и год. Верстка элемента должна быть выполнена с помощью тегов и (для выбора месяца и года). Варианты задания элемента «календарь» приведены ниже:

Вариант Язык интерфейса Формат даты

– 3: English день.месяц.год

4. Реализовать динамическую проверку корректности заполнения пользователем формы на странице «Контакт» таким образом, чтобы при потере фокуса заполняемого поля осуществлялась проверка корректности его заполнения. В случае если поле заполнено корректно, оно должно быть подсвечено зеленым цветом, иначе оно должно быть подсвечено красным, а после данного поля должна появиться надпись, поясняющая характер ошибки. После исправления пользователем ошибки, надпись должна ис83 чезнуть. Если все поля формы заполнены корректно, должна стать активной кнопка «Отправить».

5. Реализовать открытие в динамически формируемом новом окне (блоке DIV) соответствующих больших фото при щелчке мыши по маленьким фото на странице «Фотоальбом».

6. Добавить страницу «История просмотра». На данной странице реализовать отображение двух таблиц:

– «История текущего сеанса» – в данной таблице отображается количество посещений каждой страницы за время текущего сеанса. Реализовать хранение этих данных в Local Storage.
– «История за все время» – в данной таблице отображается количество посещений каждой страницы за все время. Реализовать хранение этих данных в хранилище Cookies. Для выполнения этого задания необходимо создать два JavaScript метода: getCookie (name) и setCookie(name, value, exlpiration_days).

Для реализации страницы «История просмотра» необходимо добавить на каждую страницу вызов JavaScript функции, которая будет сохранять информацию о просмотре страницы в Local Storage и Cookies.

3. Ход работы

Код на JS представлен в Приложении А.

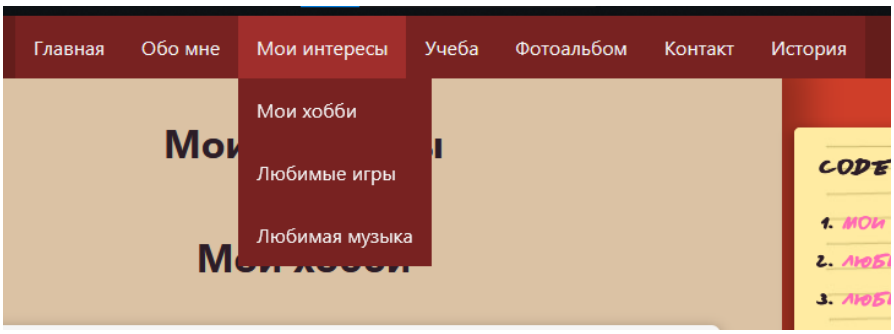

Рисунок 1 – Раскрытие меню при наведении

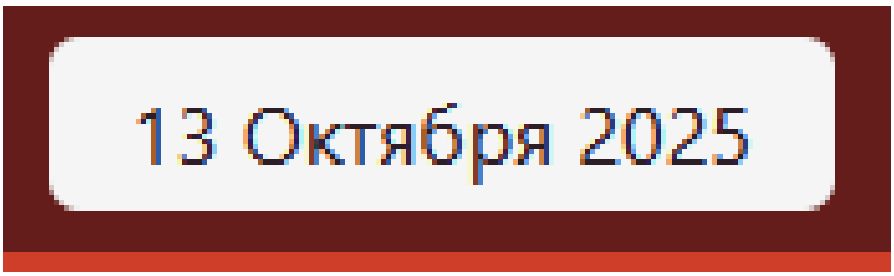

Рисунок 2 – Часы

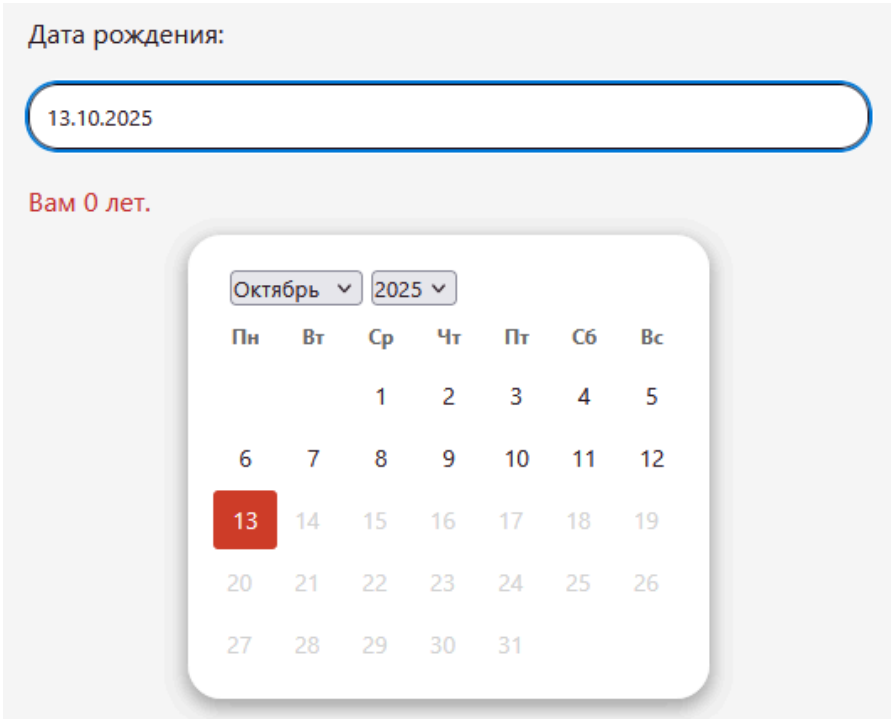

Рисунок 3 – Календарь

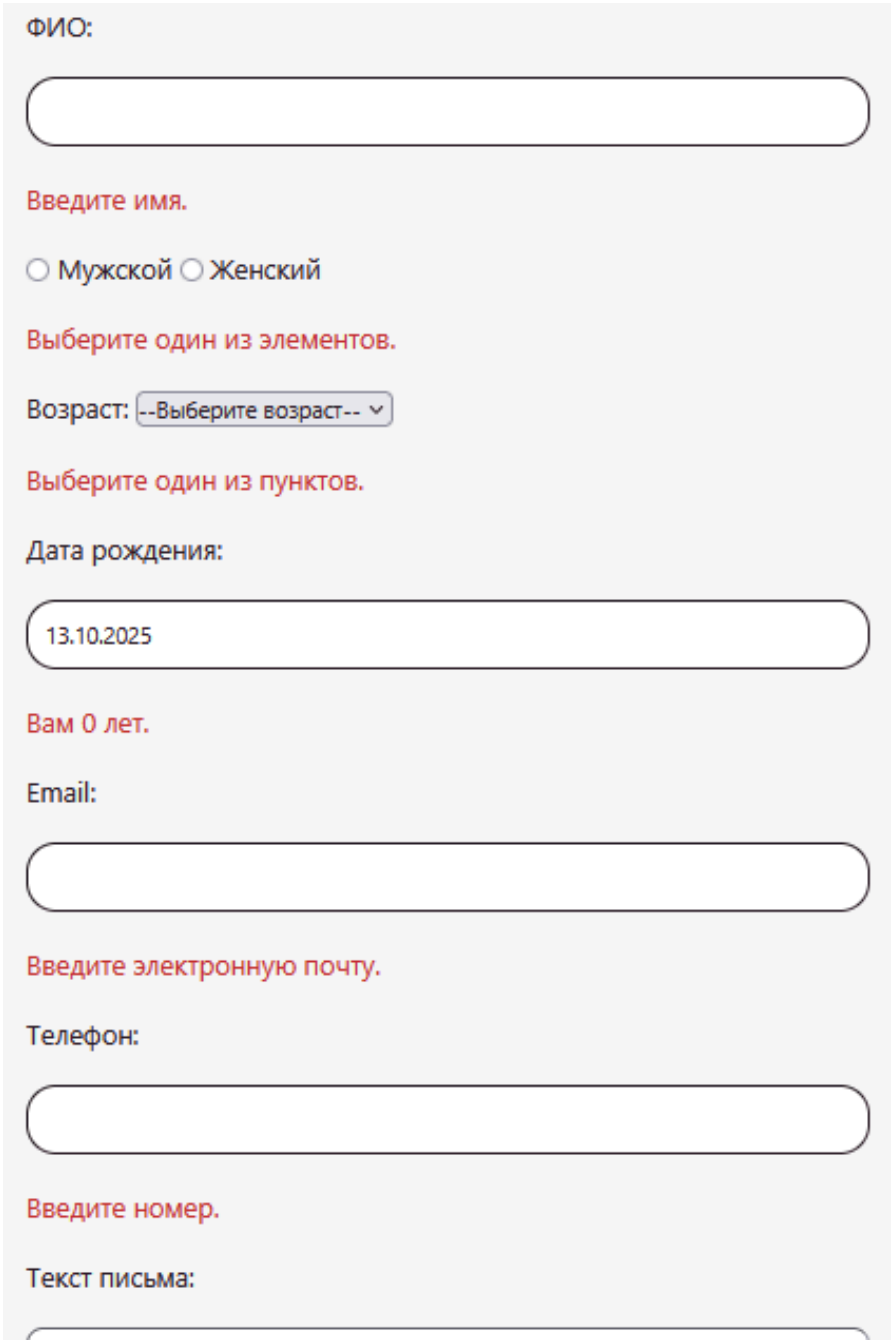

Рисунок 4 – Динамическая валидация формы

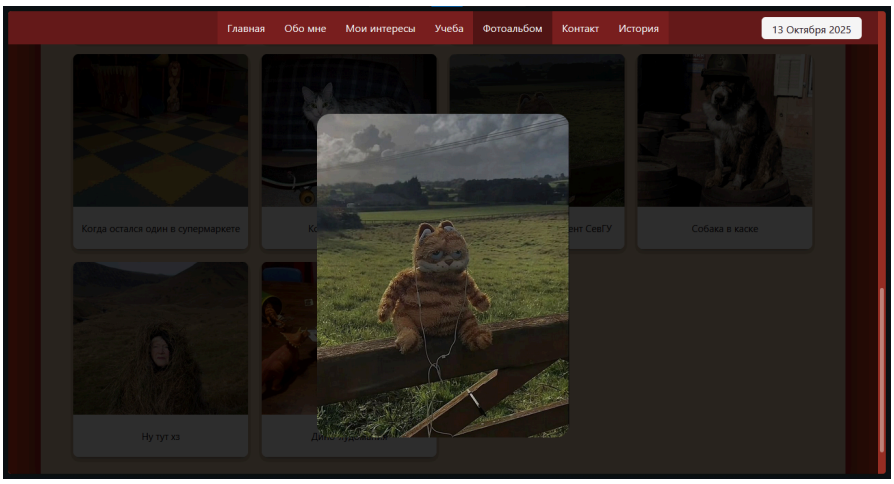

Рисунок 5 – Открытие фото в увеличенном масштабе

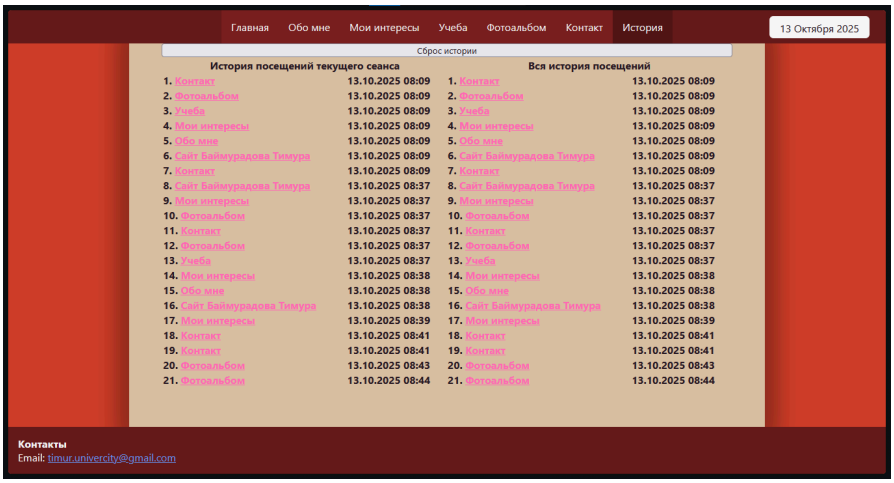

Рисунок 6 – История посещения

4. Вывод

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, Local Storage и Cookies.

В ходе выполнения работы исследована структура модули документа DOM, систему событий JS и возможность зранения данных на стороне клиента.

Для скрипта по раскрытию меню при наведении мыши использовано события mouseenter и mouseleave. Часы используют функцию setInterval для установки частоты обновления. Календарь показывается при фокусе на строку ввода даты

с помощью событий focusin и click (для скрытия календаря). Валидация формы контактов происходит на событие focusout и submit, а на reset происходит очистка ошибок. Открытие фото в увеличеном масштабе используется событие click. История посещений использует LocalStorage для зранения посещения за все время и Cookies для зранения текущей сессии. Запись посещения на страницах происходит по событию DOMContentLoaded.

ПРИЛОЖЕНИЕ А

Листинг 1 – Файл menu_reveal_on_hover.js

```
function createDropMenuItemTemplate() {
    const template = document.createElement('template');
    template.innerHTML = '<li class="drop-menu-item"><a class="nav-link"></a></li>';
    return template;
}

function appendDropMenuToElement(element, anchors) {
    const dropMenu = document.createElement('ul');
    const itemTemplate = createDropMenuItemTemplate();

    const elementRect = element.getBoundingClientRect();
    dropMenu.style.position = 'fixed';
    dropMenu.classList.add('drop-menu');
    dropMenu.style.top = `${elementRect.top + elementRect.height}px`;
    dropMenu.style.left = `${elementRect.left}px`;

    for (let i = 0; i < anchors.length; i += 1) {
        const item = document.importNode(itemTemplate.content, true);
        const a = item.querySelector('a');
        a.href = anchors[i].href;
        a.textContent = anchors[i].text;
        dropMenu.appendChild(item);
    }

    element.appendChild(dropMenu);

}

function addDropMenuEventLiseners(element, anchors) {
    element.addEventListener('mouseenter', (event) => {
        const link = event.currentTarget;
        appendDropMenuToElement(link, anchors);
    });

    element.addEventListener('mouseleave', (event) => {
        const link = event.currentTarget;
        const dropMenu = link.querySelector('.drop-menu');
        if (dropMenu != null) {
            link.removeChild(dropMenu);
        }
    });
}

const interestsLink = document.getElementById('interests-link');
addDropMenuEventLiseners(interestsLink, [
    { href: '/my_interests#hobbies', text: 'Мои хобби' },
    { href: '/my_interests#games', text: 'Любимые игры' },
    { href: '/my_interests#music', text: 'Любимая музыка' },
]);

const studesLink = document.getElementById('studies-link');
addDropMenuEventLiseners(studesLink, [
    { href: '/studies/test', text: 'Тест' },
]);
```

Листинг 2 – Файл clock.js

```
function formatClock(date) {
    const months = [
        'Января',
        'Февраля',
        'Марта',
        'Апреля',
        'Мая',
        'Июня',
        'Июля',
        'Августа',
        'Сентября',
        'Октября',
        'Ноября',
        'Декабря',
    ];
    const month = months[date.getMonth()];
    return `${date.getDate()} ${month} ${date.getFullYear()}`;
}

function updateClock(element) {
    const now = new Date();
    element.textContent = formatClock(now);
}

const clock = document.getElementById('clock');
updateClock(clock);

setInterval(() => {
    updateClock(clock);
}, 1000);
```

Листинг 3 – Файл calendar.js

```
const calendar = document.getElementById('calendar');
const input = document.getElementById('birthday-date-input');
const yearSelect = document.getElementById('year-select');
const monthSelect = document.getElementById('month-select');
let today = new Date();
let selectedDate = today;
selectDate(selectedDate);

for (let year = 1900; year <= today.getFullYear(); year += 1) {
    const yearOption = document.createElement('option');
    yearOption.value = year;
    yearOption.textContent = String(year);
    yearSelect.appendChild(yearOption);
}
yearSelect.value = today.getFullYear();

input.addEventListener('focusin', () => {
    showCalendar();
});

document.addEventListener('click', (event) => {
    if (!calendar.contains(event.target) && event.target !== input) {
        hideCalendar();
    }
});

function showCalendar() {
    calendar.classList.add('show');
```

```
    selectDate(selectedDate);
    renderCalendar(selectedDate.getFullYear(), selectedDate.getMonth());
}

function hideCalendar() {
    calendar.classList.remove('show');
}

function checkDateGreater(d1, d2) {
    const d1Time = d1.getTime();
    const d2Time = d2.getTime();
    return d1Time > d2Time;
}

function renderCalendar(year, month) {
    const firstDay = new Date(year, month, 1);
    const lastDay = new Date(year, month + 1, 0);
    const daysInMonth = lastDay.getDate();

    let firstDayOfWeek = firstDay.getDay();
    firstDayOfWeek = firstDayOfWeek === 0 ? 6 : firstDayOfWeek - 1;

    const calendarBody = document.getElementById('calendar-body');
    calendarBody.innerHTML = '';

    for (let i = 0; i < firstDayOfWeek; i += 1) {
        const blankDay = document.createElement('div');
        blankDay.className = 'blank-day';
        calendarBody.appendChild(blankDay);
    }

    for (let i = 1; i <= daysInMonth; i += 1) {
        let day = null;
        if (checkDateGreater(new Date(year, month, i), today)) {
            day = document.createElement('div');
            day.classList.add('day');
            day.classList.add('future');
            day.textContent = i;
        } else {
            day = document.createElement('button');
            day.className = 'day';
            day.textContent = i;

            if (year === today.getFullYear()
                && month === today.getMonth()
                && i === today.getDate()) {
                day.classList.add('today');
            }

            if (selectedDate
                && year === selectedDate.getFullYear()
                && month === selectedDate.getMonth()
                && i === selectedDate.getDate()) {
                day.classList.add('selected');
            }
            day.addEventListener('click', () => {
                selectDate(new Date(year, month, i));
                input.focus();
            });
        }
        calendarBody.appendChild(day);
    }

    const totalCells = 35;
    const remainingCells = totalCells - (firstDayOfWeek + daysInMonth);
    for (let i = 0; i < remainingCells; i += 1) {
```

```
        const blankDay = document.createElement('div');
        blankDay.className = 'blank-day';
        calendarBody.appendChild(blankDay);
    }
}

function selectDate(date) {
    input.value = formatDate(date);
    selectedDate = date;
    monthSelect.value = date.getMonth();
    yearSelect.value = date.getFullYear();
}

function formatDate(date) {
    const day = date.getDate().toString().padStart(2, '0');
    const month = (date.getMonth() + 1).toString().padStart(2, '0');
    const year = date.getFullYear();
    return `${day}.${month}.${year}`;
}

yearSelect.addEventListener('change', (event) => {
    const year = Number(event.currentTarget.value);
    const month = Number(monthSelect.value);
    renderCalendar(year, month);
});

monthSelect.addEventListener('change', (event) => {
    const month = Number(event.currentTarget.value);
    const year = Number(yearSelect.value);
    renderCalendar(year, month);
});
```

Листинг 4 – Файл callback_validation.js

```
function prependElement(element, prepend) {
    element.parentNode.insertBefore(prepend, element);
}

function appendElement(element, prepend) {
    element.parentNode.insertBefore(prepend, element.nextSibling);
}

function createErrorMessage(element, message) {
    const error = document.createElement('span');
    error.textContent = message;
    error.className = 'error';
    appendElement(element, error);
    return error;
}

const fullNameInput = document.getElementById('full-name');
let fullNameError = null;

fullNameInput.addEventListener('focusout', (_) => {
    validateNameInput();
});

function validateNameInput() {
    const err = checkFIO(fullNameInput.value);
    if (err != null) {
        if (fullNameError != null) {
```

```
            fullNameError.remove();
        }
        fullNameError = createErrorMessage(fullNameInput, err);
    } else {
        if (fullNameError != null) {
            fullNameError.remove();
            fullNameError = null;
        }
        return false;
    }
    return true;
}

const genderRadiosDiv = document.getElementById('gender-radios');
const genderRadios = genderRadiosDiv.querySelectorAll('input');
let genderError = null;

function validateGenderRadios() {
    let unchecked = 0;
    for (let i = 0; i < genderRadios.length; i += 1) {
        if (!genderRadios[i].checked) {
            unchecked += 1;
        }
    }
    if (genderRadios.length != 0 && unchecked == genderRadios.length) {
        if (genderError == null) {
            const br = document.createElement('br');
            const error = document.createElement('span');
            error.textContent = 'Выберите один из элементов.';
            error.className = 'error';
            appendElement(genderRadiosDiv, error);
            appendElement(error, br);
            genderError = error;
        }
        return true;
    }
    if (genderError != null) {
        genderError.nextSibling.remove();
        genderError.remove();
        genderError = null;
    }
    return false;
}

genderRadios.forEach(r => {
    r.addEventListener('change', _ => {
        validateGenderRadios();
    });
});

const ageSelect = document.getElementById('age');
let ageError = null;

function validateAgeSelect() {
    if (ageSelect.value === '') {
        if (ageError != null) {
            ageError.nextSibling.remove();
            ageError.remove();
        }
        const br = document.createElement('br');
        const error = document.createElement('span');
        error.textContent = 'Выберите один из пунктов.';
        error.className = 'error';
        appendElement(ageSelect, error);
        prependElement(error, br);
        ageError = br;
```

```
    } else {
        if (ageError != null) {
            ageError.nextSibling.remove();
            ageError.remove();
            ageError = null;
        }
        return false;
    }
    return true;
}

ageSelect.addEventListener('change', _ => {
    validateAgeSelect();
});

const birthdayInput = document.getElementById('birthday-date-input');
let birthdayError = null;

birthdayInput.addEventListener('focusout', (_) => {
    validateBirthdayInput();
});

birthdayInput.addEventListener('focusin', (_) => {
    validateBirthdayInput();
});

function validateBirthdayInput() {
    const err = checkBirthdayDate(birthdayInput.value);
    if (err != null) {
        if (birthdayError != null) {
            birthdayError.remove();
        }
        birthdayError = createErrorMessage(birthdayInput, err);
    } else {
        if (birthdayError != null) {
            birthdayError.remove();
            birthdayError = null;
        }
        return false;
    }
    return true;
}

const emailInput = document.getElementById('email');
let emailError = null;

emailInput.addEventListener('focusout', (_) => {
    validateEmailInput();
});

function validateEmailInput() {
    if (emailInput.value === '') {
        if (emailError != null) {
            emailError.remove();
        }
        emailError = createErrorMessage(emailInput, 'Введите электронную почту.');
    } else {
        if (emailError != null) {
            emailError.remove();
            emailError = null;
        }
        return false;
    }
    return true;
}
```

```
const phoneInput = document.getElementById('phone');
let phoneError = null;

phoneInput.addEventListener('focusout', (_) => {
    validatePhoneNumberInput();
});

function validatePhoneNumberInput() {
    const err = checkPhoneNumber(phoneInput.value);
    if (err != null) {
        if (phoneError != null) {
            phoneError.remove();
        }
        phoneError = createErrorMessage(phoneInput, err);
    } else {
        if (phoneError != null) {
            phoneError.remove();
            phoneError = null;
        }
        return false;
    }
    return true;
}

const textInput = document.getElementById('text');
let textError = null;

textInput.addEventListener('focusout', (_) => {
    validateTextInput();
});

function validateTextInput() {
    if (textInput.value === '') {
        if (textError != null) {
            textError.remove();
        }
        textError = createErrorMessage(textInput, 'Введите текст письма.');
    } else {
        if (textError != null) {
            textError.remove();
            textError = null;
        }
        return false;
    }
    return true;
}

const form = document.getElementById('callback-form');
form.addEventListener('submit', (e) => {
    let err = false;
    err |= validateNameInput();
    err |= validateGenderRadios();
    err |= validateAgeSelect();
    err |= validateBirthdayInput();
    err |= validateEmailInput();
    err |= validatePhoneNumberInput();
    err |= validateTextInput();
    if (err) {
        e.preventDefault();
        return false;
    }
    return true;
}, false);

form.addEventListener('reset', _ => {
    if (fullNameError != null) {
```

```
        fullNameError.remove();
        fullNameError = null;
    }
    if (genderError != null) {
        genderError.nextSibling.remove();
        genderError.remove();
        genderError = null;
    }
    if (ageError != null) {
        ageError.nextSibling.remove();
        ageError.remove();
        ageError = null;
    }
    if (birthdayError != null) {
        birthdayError.remove();
        birthdayError = null;
    }
    if (emailError != null) {
        emailError.remove();
        emailError = null;
    }
    if (phoneError != null) {
        phoneError.remove();
        phoneError = null;
    }
    if (textError != null) {
        textError.remove();
        textError = null;
    }

});

function checkFIO(fio) {
    if (fio === '') {
        return 'Введите имя.';
    }
    const words = fio.split(' ');
    if (words.length !== 3) {
        return 'Фио должно иметь 3 слова.';
    }
    return null;
}

function checkPhoneNumber(phoneNumber) {
    if (phoneNumber.length === 0) {
        return 'Введите номер.';
    }

          if (phoneNumber.charAt(0) !== '+' || (phoneNumber.charAt(1) !== '7' &&
phoneNumber.charAt(1) !== '3')) {
        return 'Номер телефона должен начинать с +7 или +3.';
    }

    let digitCount = 0;
    let hasWhiteSpace = false;
    let otherCount = 0;
    for (let i = 1; i < phoneNumber.length; i += 1) {
        const ch = phoneNumber.charAt(i);
        if (ch >= '0' && ch <= 9) {
            digitCount += 1;
        } else if (ch === ' ' || ch === '\n' || ch === '\t' || ch === '\r') {
            hasWhiteSpace = true;
        } else {
            otherCount += 1;
        }
    }
```

```
    if (hasWhiteSpace) {
        return 'Номер телефона не должен иметь пробелов.';
    }

    if (otherCount !== 0) {
        return 'Номер телефона может иметь только цифры и символ \'+\'.';
    }

    if (digitCount < 9 || digitCount > 11) {
        return 'Номер телефона должен иметь от 9 до 11 цифр.';
    }

    return null;
}

function checkBirthdayDate(date) {
    if (date === '') {
        return 'Выберите дату.';
    }
    const today = new Date();
   const tomorrow = new Date(today.getFullYear(), today.getMonth(), today.getDate()).getTime();
    const [day, month, year] = date.split('.');
    const pickedDate = new Date(year, month - 1, day).getTime();
    if (pickedDate > tomorrow) {
        return 'Выберите дату в прошлом.';
    } else if (pickedDate == tomorrow) {
        return 'Вам 0 лет.';
    }
    return null;
}
```

Листинг 5 – Файл photoalbum.js

```
const photos = [
     { filename: "photo1.jpg", alt: "Горящее пианино", title: "Это Imagine Dragons?", label:
"Огненная музыка" },
     { filename: "photo2.jpg", alt: "Волынщик на моноколесе", title: "HELL YEAH!!!", label:
"Эпичный волынщик" },
     { filename: "photo3.jpg", alt: "Гаст из майнкрафта с атомной бомбой", title: "Надеюсь
ничего не произойдет", label: "Гаст с царь бимбой" },
    { filename: "photo4.jpg", alt: "Лягушка со скрипкой", title: "Это среда чувакиииииииии!",
label: "Лягушка" },
   { filename: "photo5.jpg", alt: "Круглый пруд", title: "Везде обман! Торт это ЛОЖЬ!", label:
"Круглый пруд" },
    { filename: "photo6.jpg", alt: "Корова с лестницей на голове", title: "Любопытной корове
вымя оторвали", label: "Любопытная корова" },
   { filename: "photo7.jpg", alt: "Мужик с крокодилом", title: "Это могли бы быть мы с тобой,
но ты не крокодил в очках", label: "Мужик с крокодилом" },
   { filename: "photo8.jpg", alt: "Страшные костюмы", title: "Я и мои последние две извилины",
label: "Ночной кошмар" },
  { filename: "photo9.jpg", alt: "AAAAAAAAAAAAAAA!!!", title: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
label: "AAAAAAAAAAAAAA!!!!!!" },
     { filename: "photo10.jpg", alt: "Младенец с РПГ", title: "Я не знаю что тут дополнять",
label: "Идеальный солдат" },
    { filename: "photo11.jpg", alt: "Круто!", title: "Круто!", label: "Круто!" },
     { filename: "photo12.jpg", alt: "Японец с огненным оружием", title: "Эпично!", label:
"Огненная крутилка" },
   { filename: "photo13.jpg", alt: "Закрытая на ночь игровая площадка с супермаркете", title:
"Обернись", label: "Когда остался один в супермаркете" },
      { filename: "photo14.jpg", alt: "Кот на скейтборде", title: "Крутой!", label: "Кот
на скейтборде" },
     { filename: "photo15.jpg", alt: "Депресивный Гарфилд", title: "Тупа я", label: "Самый
```

```
счастливый студент СевГУ" },
     { filename: "photo16.jpg", alt: "Собака в каске", title: "Старое фото", label: "Собака
в каске" },
    { filename: "photo17.jpg", alt: "Бабка в костюме из травы", title: "Фото холмов", label:
"Ну тут хз" },
    { filename: "photo18.jpg", alt: "Динозавры-лудоманы", title: "Let's go gambling!", label:
"Дино-лудомания" }
];

const photoAlbum = document.querySelector(".photo-container");
const fragment = document.createDocumentFragment();
const template = document.querySelector("#photo-card-template");

for (let i = 0; i < photos.length; i += 1) {
    const { filename, alt, title, label } = photos[i];
    const card = document.importNode(template.content, true);

    const imageContainer = card.querySelector(".photo-image-container");
    const img = document.createElement("img");
    img.src = `/media/photo/${filename}`;
    img.alt = alt;
    img.title = title;
    imageContainer.appendChild(img);

    const labelContainer = card.querySelector(".photo-label");
    const p = document.createElement("p");
    p.textContent = label;
    labelContainer.appendChild(p);

    fragment.appendChild(card);
}

photoAlbum.appendChild(fragment);

const fullscreenDiv = document.getElementById('fullscreen-photo');
let fullscreenOpen = false;
function openFullscreen(src) {
    const img = document.createElement('img');
    img.src = src;
    img.className = 'rounded';
    fullscreenDiv.appendChild(img);
    fullscreenDiv.classList.add('show');
    fullscreenOpen = true;
}

const photoCards = document.querySelectorAll('.photo-card');
photoCards.forEach(card => card.addEventListener('click', e => {
    const img = e.currentTarget.querySelector('img');
    openFullscreen(img.src);
}));

fullscreenDiv.addEventListener('click', e => {
    if (fullscreenOpen) {
        const img = fullscreenDiv.querySelector('img');
        if (e.target !== img) {
            img.remove();
            fullscreenDiv.classList.remove('show');
            fullscreenOpen = false;
        }
    }
})
```

Листинг 6 – Файл history.js

```
function getCookies() {
    if (document.cookie === '') {
        return {
            pageNames: '',
            pageLinks: '',
            times: '',
        }
    }
    const cookies = {};
    document.cookie.split('; ').forEach(c => {
        const [key, value] = c.split('=');
        cookies[key] = value;
    });
    return cookies;
}

function getSessionHistoryFromCookies() {
    const cookies = getCookies();
    const pageNames = cookies.pageNames.split('&');
    const pageLinks = cookies.pageLinks.split('&');
    const times = cookies.times.split('&');

    if (pageNames[0] === ''
        || pageLinks[0] === ''
        || times[0] === '') {
        return [];
    }

    const history = [];
    for (let i = 0; i < pageNames.length; i += 1) {
        history.push({
            pageName: pageNames[i],
            pageLink: pageLinks[i],
            time: times[i],
        });
    }
    return history;
}

function setSessionHistoryInCookies(history) {
    let pageNames = history[0].pageName;
    let pageLinks = history[0].pageLink;
    let times = history[0].time;

    for (let i = 1; i < history.length; i += 1) {
        pageNames += '&' + history[i].pageName;
        pageLinks += '&' + history[i].pageLink;
        times += '&' + history[i].time;
    }

    document.cookie = `pageNames=${pageNames}`;
    document.cookie = `pageLinks=${pageLinks}`;
    document.cookie = `times=${times}`;
    document.cookie = `Max-Age=${60 * 60}`;
}

function getAllTimeHistoryFromLS() {
    const localStorageItem = localStorage.getItem('allTimeHistory');
    if (localStorageItem == null) {
        return [];
    }
    return JSON.parse(localStorageItem);
}

function setAllTimeHistoryInLS(history) {
    localStorage.setItem('allTimeHistory', JSON.stringify(history));
```

```
}

function formatDate(date) {
    const year = date.getFullYear().toString().padStart(2, '0');
    const month = (date.getMonth() + 1).toString().padStart(2, '0');
    const day = date.getDate().toString().padStart(2, '0');
    const hour = date.getHours().toString().padStart(2, '0');
    const minute = date.getMinutes().toString().padStart(2, '0');
    return `${day}.${month}.${year} ${hour}:${minute}`;
}

function max(a, b) {
    if (a >= b) {
        return a;
    } else {
        return b;
    }
}

function parseDate(dateStr) {
    const [date, time] = dateStr.split(' ');
    const [day, month, year] = date.split('.');
    const [hour, minute] = time.split(':');
    return new Date(year, month-1, day, hour, minute);
}

function displayHistory() {
    const allTimeHistory = getAllTimeHistoryFromLS();
    const sessionHistory = getSessionHistoryFromCookies();

    const historyTable = document.getElementById('history-table');
    let html = '';

    let maxLength = max(sessionHistory.length, allTimeHistory.length);
    for (let i = 0; i < maxLength; i += 1) {
        html += '<tr>'

        if (i < sessionHistory.length) {
                                html += `<th style="text-align:left">${i + 1}. <a
href="${sessionHistory[i].pageLink}">${sessionHistory[i].pageName}</a></th>
                <th style="text-align:left">${formatDate(parseDate(sessionHistory[i].time))}
</th>`;
        } else {
            html += '<th></th><th></th>';
        }

        if (i < allTimeHistory.length) {
                                html += `<th style="text-align:left">${i + 1}. <a
href="${allTimeHistory[i].pageLink}">${allTimeHistory[i].pageName}</a></th>
                <th style="text-align:left">${formatDate(parseDate(allTimeHistory[i].time))}
</th>`;
        } else {
            html += '<th></th><th></th>';
        }

        html += '</tr>'
    }

    historyTable.innerHTML = html;
}

function trackPage(pageName, pageLink) {
    const allTimeHistory = getAllTimeHistoryFromLS();
    const sessionHistory = getSessionHistoryFromCookies();
    let now = formatDate(new Date());
```

```
    if (sessionHistory.length >= 121) {
        for (let i = 0; i < sessionHistory.length - 1; i += 1) {
            sessionHistory[i] = sessionHistory[i + 1];
        }
        sessionHistory[120] = {
            pageName: pageName,
            pageLink: pageLink,
            time: now,
        };
    } else {
        sessionHistory.push({
            pageName: pageName,
            pageLink: pageLink,
            time: now,
        });
    }

    allTimeHistory.push({
        pageName: pageName,
        pageLink: pageLink,
        time: now,
    });

    setSessionHistoryInCookies(sessionHistory);
    setAllTimeHistoryInLS(allTimeHistory);
}

function resetHistory() {
    setSessionHistoryInCookies([{
        pageName: '',
        pageLink: '',
        time: '',
    }]);
    setAllTimeHistoryInLS([]);
    const historyTable = document.getElementById('history-table');
    historyTable.innerHTML = '';
}

document.addEventListener('DOMContentLoaded', _ => {
    const pathDecomposed = window.location.href.split('/');
    if (pathDecomposed[pathDecomposed.length - 1] === 'history') {
        return;
    }
    const href = '/' + pathDecomposed[pathDecomposed.length - 1];
    trackPage(document.title, href);
});
```