Министерство науки и высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №3
«Исследование объектной модели документа (DOM) и системы событий JavaScript»
по дисциплине
«ВЕБ-ТЕХНОЛОГИИ»

Выполнил студент группы ИС/б-17-2-о
Горбенко К. Н.
Проверил
Дрозин А.Ю.

Севастополь
2019

1 ЦЕЛЬ РАБОТЫ

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 ЗАДАНИЕ НА РАБОТУ

1. Реализовать отображение в области меню сайта текущих даты и времени (обновление времени 1 раз в секунду).

2. На странице «Контакт» добавить поле «Дата рождения», для которого реализовать всплывающий снизу элемент «календарь».

3. 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 исчезнуть. Если все поля формы заполнены корректно должна стать активной кнопка «Отправить».

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

5. Добавить страницу «История просмотра». На данной странице реализовать отображение двух таблиц: «История текущего сеанса», «История за все время».

3 ХОД РАБОТЫ

3.1 Дата и время в меню сайта

Реализуем отображение в меню сайта текущих даты и времени. Для этого добавим модуль следующего содержания:

```
1 export const formatDate = (date: Date): string => {
2     return `${date.getDate()}.` +
```

```
            `${date.getMonth() + 1}.` +
            `${date.getFullYear()}, ` +
            `${date.toLocaleString(window.navigator.language, { weekday: 'long
                '})}`;
};

export const formatTime = (time : Date): string => {
    return time.toLocaleString(window.navigator.language, { hour: '2-digit',
        minute: '2-digit', second: '2-digit' });
};

const formatCurrentDate = () => {
    return formatDate(new Date());
};

const formatCurrentTime = (): string => {
    return formatTime(new Date());
};

export const updateClockOnInterval = (dateElement: HTMLElement, timeElement:
    HTMLElement, interval: number) => {
    dateElement.innerHTML = formatCurrentDate();
    timeElement.innerHTML = formatCurrentTime();
    setInterval(() => {
        dateElement.innerHTML = formatCurrentDate();
        timeElement.innerHTML = formatCurrentTime();
    }, interval);
};
```

Используется данный модуль следующим образом:

```
import { updateClockOnInterval } from '../clock/clock';

window.onload = () => {
    updateClockOnInterval(document.getElementById('date'), document.
        getElementById('time'), 1000);
};
```

Добавим в HTML разметку новые элементы:

```
<nav>
    <ul>
        <li><span                    >Web lab1</span></li>
        <li><a href="index.html"    >Home</a></li>
        <li><a href="about.html"    >About</a></li>
        <li><a href="interests.html">My interests</a></li>
        <li><a href="learning.html" >Learning</a></li>
        <li><a href="photos.html"   >Photos</a></li>
        <li><a href="contact.html"  >Contact</a></li>
```

```
10          <li><a href="test.html"      >Test</a></li>
11          <li><a href="history.html"   >History</a></li>
12          <li><span id="time"></span><br><span id="date"></span></li>
13      </ul>
14  </nav>
```

Результат изображен на рисунке 1:

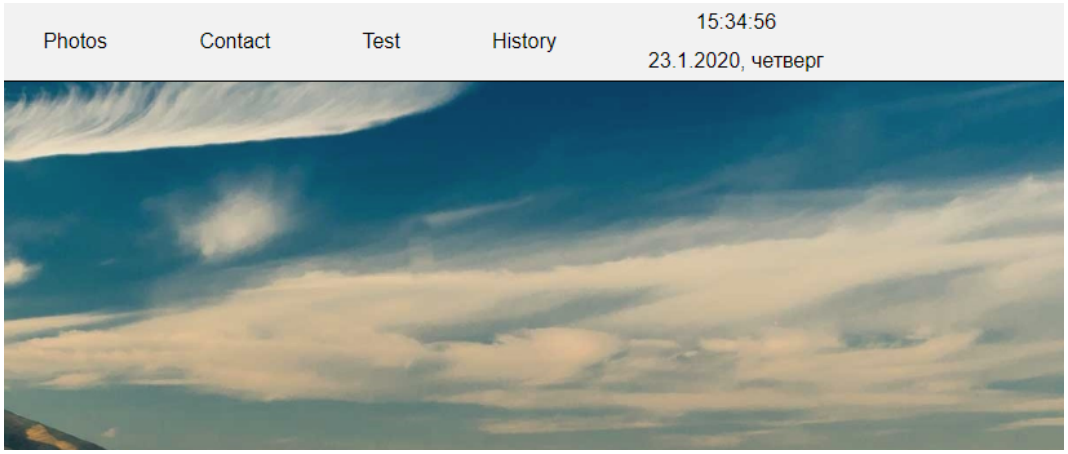


Рисунок 1 – Дата и время в меню сайта

3.2 Мои интересы

На странице «Мои интересы» реализуем выведение содержания по нажатию на соответствующий пункт меню:

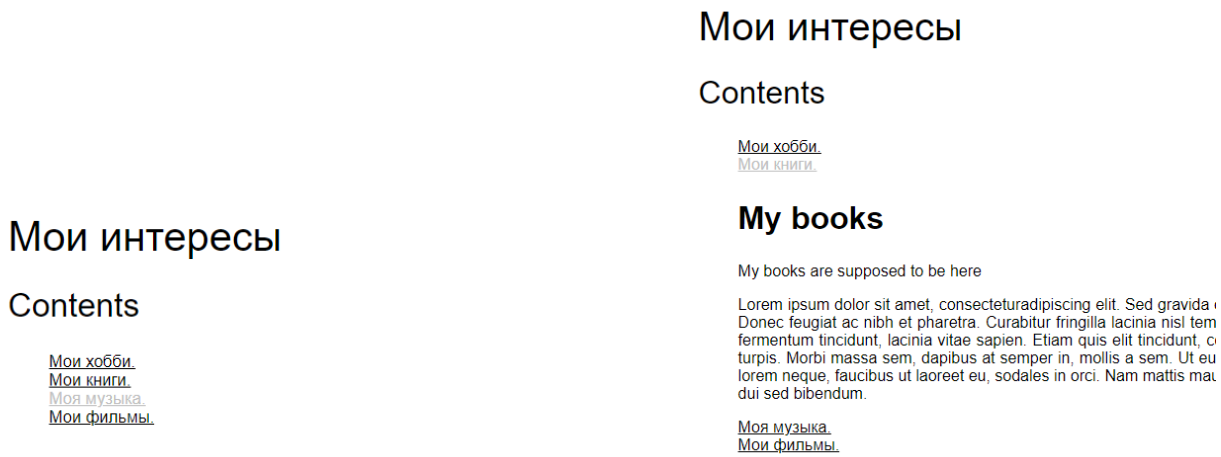
```
1  enum InterestIds {
2      Hobbies = 'Hobbies',
3      Books   = 'Books',
4      Music   = 'Music',
5      Films   = 'Films'
6  }
7
8  type InterestsNode = {
9      [key in keyof typeof InterestIds]: InterestsNodeContent
10 }
11
12 interface InterestsNodeContent {
13     header: string;
14     values: string[];
15     isHidden: boolean;
16 }
17
18 window.onload = () => {
19     updateClockOnInterval(document.getElementById('date'), document.
           getElementById('time'), 1000);
```

```
20      let interestsLinks = document.getElementsByClassName('contents');
21
22      Array.from(interestsLinks).forEach(link => {
23          link.addEventListener('click', handleClick);
24      });
25  };
26
27  const handleClick = (event) : void => {
28      const target = event.target as HTMLElement;
29      if (interests[target.id].isHidden) {
30          insertAfter(createInterestNode(target.id), target);
31          interests[target.id].isHidden = false;
32      } else {
33          removeElementById(`${target.id}-expand`);
34          interests[target.id].isHidden = true;
35      }
36  };
37
38  const createInterestNode = (interestId: string) => {
39      const interestContent = interests[interestId];
40      const interestNode = createElement('li', { id: `${interestId}-expand`,
            classList: ['interests'] });
41      appendChildren(interestNode,
42          createElement('h2', { innerHTML: interestContent.header }),
43          createElement('p', { innerHTML: interestContent.values[0] }),
44          createElement('p', { innerHTML: interestContent.values[1] })
45      );
46      return interestNode;
47  };
```

Обновим HTML документ:

```
1 <section>
2 <header><h2>Contents</h1></header>
3 <ul id="interests">
4     <li class="contents" id="Hobbies">Мои хобби.</li>
5     <li class="contents" id="Books">Мои книги.</li>
6     <li class="contents" id="Music">Моя музыка.</li>
7     <li class="contents" id="Films">Мои фильмы.</li>
8 </ul>
9 </section>
```

Меню выглядит следующим образом (рисунок 2):



(a) До нажатия (b) После нажатия

Рисунок 2 – Выпадающее меню на странице «Мои интересы»

3.3 Формы

3.3.1 Выбор даты

Для реализации выбора даты создадим отдельный модуль. Его содержание:

```
1 import { createElementWithClass, createElementWithInnerHTML,
      createElementWithId } from '../utils/dom';
2 import { formatDate } from '../clock/clock';
3
4 const datepickerId = 'datepicker';
5 const datepickerInputId = 'datepicker-input';
6 const datepickerYearSelectId = 'datepicker-year';
7 const datepickerMonthSelectId = 'datepicker-month';
8 const datepickerDateListId = 'datepicker-date-list';
9 const elementWrapperClass = 'element-wrapper';
10
11 const datepicker = () => document.querySelector('#' + datepickerId);
12 const datepickerInput = () => document.querySelector('#' + datepickerInputId);
13 const yearSelect = () => document.querySelector('#' + datepickerYearSelectId);
14 const monthSelect = () => document.querySelector('#' + datepickerMonthSelectId)
      ;
15 const dateList = () => document.querySelector('#' + datepickerDateListId);
16
17 let selectedYear, selectedMonthNumber, selectedDate;
18 let isShown = false;
19
20 export default () => {
21     datepickerInput().addEventListener('click', showDatePicker);
22     window.addEventListener('click', (event) => removeDatePicker(event));
23 }
```

```
24
25 const insertDate = (event) => {
26     if (event.target !== dateList()) {
27         selectedYear = (yearSelect() as HTMLFormElement).value;
28         selectedMonthNumber = months.indexOf((monthSelect() as HTMLFormElement)
               .value);
29         selectedDate = event.target.innerHTML;
30         const date = new Date(selectedYear, selectedMonthNumber, selectedDate);
31
32         (datepickerInput() as HTMLElement).focus();
33         (datepickerInput() as HTMLFormElement).value = formatDate(date);
34         (datepickerInput() as HTMLElement).blur();
35     }
36 };
37
38 const showDatePicker = () => {
39     if (!isShown) {
40         datepickerInput().parentNode.insertBefore(createDatePicker(),
               datepickerInput().nextSibling);
41         Array.from(dateList().childNodes).forEach(listItem => {
42             listItem.addEventListener('click', (event) => insertDate(event));
43         });
44         yearSelect().addEventListener('change', updateDatesList);
45         monthSelect().addEventListener('change', updateDatesList);
46         isShown = true;
47     }
48 };
49
50 const removeDatePicker = (event) => {
51     if (isShown && !datepicker().contains(event.target as Node) && event.target
           !== datepickerInput()) {
52         (datepickerInput() as HTMLElement).blur();
53         datepicker().remove();
54         isShown = false;
55     }
56 };
57
58 const createDatePicker = () => {
59     const date = new Date();
60     let datepicker = createElementWithId('div', datepickerId);
61     let yearAndMonthSelectsWrapper = createElementWithClass('div',
           elementWrapperClass);
62     let dateListWrapper = createElementWithClass('div', elementWrapperClass);
63     yearAndMonthSelectsWrapper.appendChild(createYearsSelect());
64     yearAndMonthSelectsWrapper.appendChild(createMonthSelect());
65     datepicker.appendChild(yearAndMonthSelectsWrapper);
66     dateListWrapper.appendChild(createDatesList(selectedYear || date.
           getFullYear(), selectedMonthNumber + 1 || date.getMonth() - 1));
```

```
67      datepicker.appendChild(dateListWrapper);
68
69      return datepicker;
70  };
71
72  const updateDatesList = () => {
73      const selectedYear = (yearSelect() as HTMLFormElement).value;
74      const selectedMonthNumber = months.indexOf((monthSelect() as
            HTMLFormElement).value);
75      const dateListWrapper = dateList().parentElement;
76
77      dateList().remove();
78      const newDateList = createDatesList(selectedYear, selectedMonthNumber + 1);
79      newDateList.addEventListener('click', (event) => insertDate(event));
80      dateListWrapper.appendChild(newDateList);
81  };
82
83  const createYearsSelect = () => {
84      const selectYear = createElementWithId('select', datepickerYearSelectId);
85      getYears().forEach(year => {
86          selectYear.appendChild(createElementWithInnerHTML('option', year));
87      });
88      (selectYear as HTMLFormElement).value = selectedYear || new Date().
            getFullYear();
89      return selectYear;
90  };
91
92  const createMonthSelect = () => {
93      const selectMonth = createElementWithId('select', datepickerMonthSelectId);
94      months.forEach(month => {
95          selectMonth.appendChild(createElementWithInnerHTML('option', month));
96      });
97      (selectMonth as HTMLFormElement).value = months[selectedMonthNumber] ||
            months[new Date().getMonth()];
98      return selectMonth;
99  };
100
101 const createDatesList = (year, month) => {
102     let dateList = createElementWithId('ul', datepickerDateListId);
103     getDays(getNumberOfDaysInMonth(year, month)).forEach(date => {
104         dateList.appendChild(createElementWithInnerHTML('li', date));
105     });
106     return dateList;
107 };
108
109 const getNumberOfDaysInMonth = (year, month) => {
110     return new Date(year, month, 0).getDate();
111 };
```

```
112
113 const getYears = () => {
114     let years = [];
115     for (let i = 2000; i <= 2025; i++) {
116         years.push(i);
117     }
118     return years;
119 };
120
121 const getDays = (n : number) => {
122     let days = [];
123     for (let i = 1; i <= n; i++) {
124         days.push(i);
125     }
126     return days;
127 };
128
129 const months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
        'August', 'September', 'October', 'November', 'December'];
```

Используем этот элемент:

```
1 <li>
2     <label for="datepicker-input">Date</label>
3     <input type="text" name="date" id="datepicker-input" class="field-long" />
4 </li>
```

Настроим валидацию для этого элемента:

```
1 import { FormComponent, NameValidator, FieldFilledValidator,
      PhoneNumberValidator, setFieldsForValidation, DateValidator } from '../
      forms/forms';
2
3 window.onload = () => {
4     datepicker();
5     setFieldsForValidation(fields, document.forms.item(0));
6 };
7
8 const fields: FormComponent[] = [
9     ...
10    new FormComponent(
11        'datepicker-input',
12        [
13            new DateValidator()
14        ]
15    )
16 ];
```

Класс `DateValidator` из модуля `forms`:

```
export class DateValidator implements FormValidator {
    errorMessage = 'Date was not in correct format';
    validate = (value: string) : Boolean => {
        const pattern = /^\d{1,2}\.\d{1,2}\.\d{4}, [а-яА-Я]+$/;
        return pattern.test(value);
    };
}
```

Элемент изображен на рисунке 3.

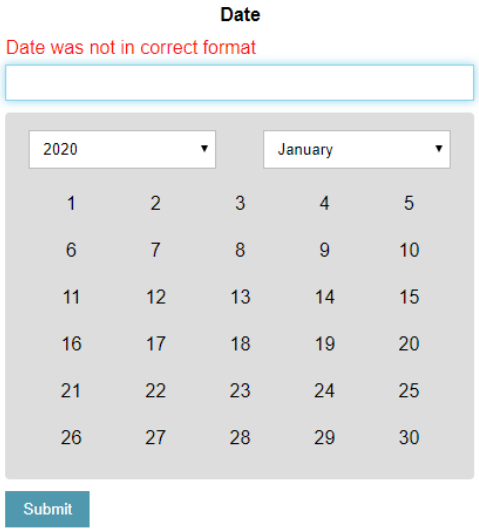


Рисунок 3 – Элемент для выбора даты и времени

3.4 Просмотр фотографий

Реализуем просмотр фотографий по нажатию. Логика реализована в соответствующем модуле.

```
import { createElement } from '../utils/dom'

const lightboxId = 'lightbox';
const lightboxClass = 'lightbox';
const lightboxPictureClass = 'lightbox-picture';
const lightboxCloseButtonClass = 'lightbox-close';

export default (lightboxPhotosWrapper: Element) => {
    lightboxPhotosWrapper.addEventListener('click', (event) => expandLightbox(
        event));

    window.onclick = (event) => {
        if (event.target === document.getElementById(lightboxId)) {
            collapseLightbox();
        }
```

```
15      }
16  }
17
18  const expandLightbox = (event) => {
19      const lightbox = createElement('div', { id: lightboxId, classList: [
            lightboxClass] });
20      const image = createElement('img', { src: event.target.src, classList: [
            lightboxPictureClass] });
21      const closeButton = createElement('div', { innerHTML: '&times', classList:
            [lightboxCloseButtonClass], onclick: () => collapseLightbox() });
22
23      lightbox.appendChild(image);
24      lightbox.appendChild(closeButton);
25      document.body.appendChild(lightbox);
26  };
27
28  const collapseLightbox = () => {
29      const lightbox = document.getElementById(lightboxId);
30      lightbox.remove();
31  };
```

Использование модуля:

```
1  import addLightbox from '../lightbox/lightbox';
2
3  window.onload = () => {
4      let photoWrapper = document.querySelector('.photo-wrapper');
5      addLightbox(photoWrapper);
6  };
```

Элемент изображен на рисунке 4.

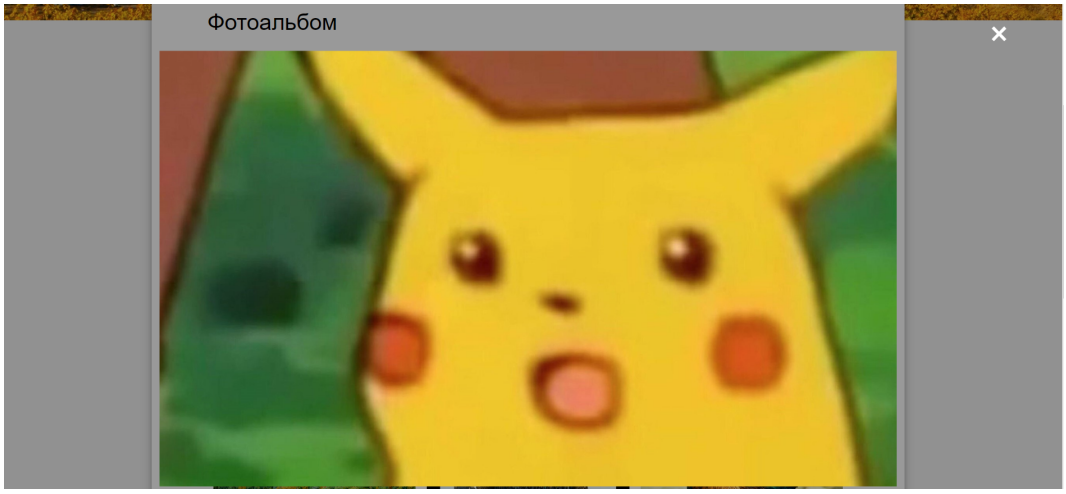


Рисунок 4 – Элемент для выбора даты и времени

3.5 История просмотра

Реализация истории просмотра в отдельном модуле:

```
type Pages = 'home'
    | 'about'
    | 'interests'
    | 'learning'
    | 'photos'
    | 'contact'
    | 'test'
    | 'history';

type PageHistory = {[key in Pages]: number;};

export const visitPage = (page: Pages) => {
    incrementLocalStoragePageVisits(page);
    incrementSessionStoragePageVisits(page);
};

const incrementLocalStoragePageVisits = (page: Pages) => {
    const currentLocalStorageValue = getLocalStorageValue(page);
    if (currentLocalStorageValue) {
        setLocalStorageValue(page, (Number(currentLocalStorageValue) + 1).
            toString());
    } else {
        setLocalStorageValue(page, '1');
    }
};

const incrementSessionStoragePageVisits = (page: Pages) => {
    const currentSessionStorageValue = getSessionStorageValue(page);
    if (currentSessionStorageValue) {
        setSessionStorageValue(page, (Number(currentSessionStorageValue) + 1).
            toString());
    } else {
        setSessionStorageValue(page, '1');
    }
};

export const getGlobalHistory = (): PageHistory => {
    return {
        home: Number(getLocalStorageValue('home')),
        about: Number(getLocalStorageValue('about')),
        interests: Number(getLocalStorageValue('interests')),
        learning: Number(getLocalStorageValue('learning')),
        photos: Number(getLocalStorageValue('photos')),
        contact: Number(getLocalStorageValue('contact')),
        test: Number(getLocalStorageValue('test')),
```

```
44          history: Number(getLocalStorageValue('history'))
45      };
46  };
47
48  export const getSessionHistory = () => {
49      return {
50          home: Number(getSessionStorageValue('home')),
51          about: Number(getSessionStorageValue('about')),
52          interests: Number(getSessionStorageValue('interests')),
53          learning: Number(getSessionStorageValue('learning')),
54          photos: Number(getSessionStorageValue('photos')),
55          contact: Number(getSessionStorageValue('contact')),
56          test: Number(getSessionStorageValue('test')),
57          history: Number(getSessionStorageValue('history'))
58      };
59  };
60
61  export const getLocalStorageValue = (name: string) => {
62      return localStorage.getItem(name);
63  };
64
65  export const setLocalStorageValue = (name: string, value: string) => {
66      localStorage.setItem(name, value);
67  };
68
69  export const getSessionStorageValue = (name: string) => {
70      return sessionStorage.getItem(name);
71  };
72
73  export const setSessionStorageValue = (name: string, value: string) => {
74      sessionStorage.setItem(name, value);
75  };
```

Поместим на каждую страницу логику сохранения информации о посещениях:

```
1  import { visitPage } from '../storage/storage';
2
3  window.onload = () => {
4      visitPage('home');
5  };
```

Реализуем страницу истории посещений:

```
1  <div class="main-content">
2      <header><h1>История</h1></header>
3      <section>
4          <header><h2>История сессии</h2></header>
5          <table class="session-history">
```

```
            <tr><th colspan="2"> История сессии</th></tr>
            <tr><td>Домашняя страница</td><td class='home'></td></tr>
            <tr><td>Обо мне</td><td class='about'></td></tr>
            <tr><td>Мои интересы</td><td class='interests'></td></tr>
            <tr><td>Учеба</td><td class='learning'></td></tr>
            <tr><td>Фото</td><td class='photos'></td></tr>
            <tr><td>Контакт</td><td class='contact'></td></tr>
            <tr><td>Тест</td><td class='test'></td></tr>
            <tr><td>История</td><td class='history'></td></tr>
        </table>
    </section>
    <section>
        <header><h2>История за все время</h2></header>
        <table class="global-history">
            <tr><th colspan="2">История за все время</th></tr>
            <tr><td>Домашняя страница</td><td class='home'></td></tr>
            <tr><td>Обо мне</td><td class='about'></td></tr>
            <tr><td>Мои интересы</td><td class='interests'></td></tr>
            <tr><td>Учеба</td><td class='learning'></td></tr>
            <tr><td>Фото</td><td class='photos'></td></tr>
            <tr><td>Контакт</td><td class='contact'></td></tr>
            <tr><td>Тест</td><td class='test'></td></tr>
            <tr><td>История</td><td class='history'></td></tr>
        </table>
    </section>
</div>
```

```
import { getGlobalHistory, getSessionHistory, visitPage } from '../storage/
    storage';

window.onload = () => {
    visitPage('history');
    const globalHistory = getGlobalHistory();
    for (let key in globalHistory) {
        const pageHistoryCell = document.querySelector(`.global-history .${key
            }`);
        pageHistoryCell.innerHTML = globalHistory[key];
    }
    const sessionHistory = getSessionHistory();
    for (let key in sessionHistory) {
        const pageHistoryCell = document.querySelector(`.session-history .${key
            }`);
        pageHistoryCell.innerHTML = sessionHistory[key];
    }
};
```

Страница истории просмотров имеет вид, изображенный на рисунках 5 - 6.

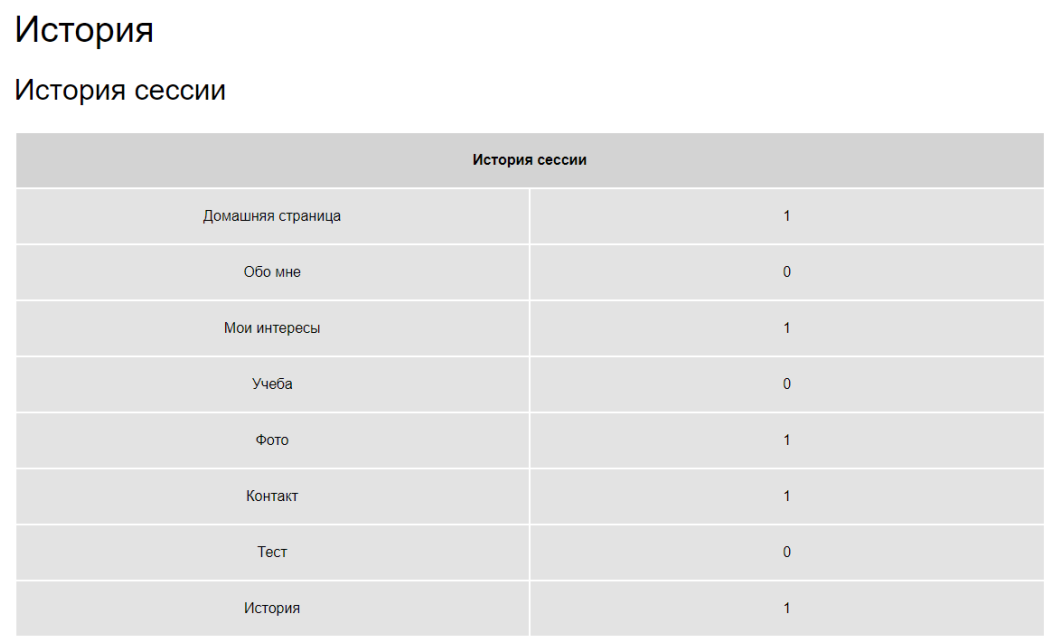


Рисунок 5 – История сессии

Рисунок 6 – История за все время

ВЫВОД

В ходе лабораторной работы активно использовался DOM API. Для получения элементов DOM можно использовать функции `getElementById`, `getElementsByCla` `querySelector`. Для размещения объектов `JavaScript` в DOM используются функции `appendChild`, `insertBefore`.

Недостатком DOM API является многословность при выполнении даже ба-

зовых операций, что приводит к тому, что код сложно поддерживать. Кроме того, отсутствует поддержка привязки данных к объектам DOM.