МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра технологий программирования

Кот Максим Артурович
Ларионова Анастасия Александровна
Щербаченя Михаил Евгеньевич

Технологии программирования

Отчет по лабораторной работе №7

«Работа с DOM»

студентов 3 курса 8 группы

Преподаватель
Терех Владимир Сергеевич

Минск, 2024

Описание предметной области

АИС «ЖД вокзал покупка билетов на поезд» — это система, которая предназначена для автоматизации процесса выбора билетов на пригородные, междугородние и международные рейсы поездов, упрощения процесса их бронирования, оплаты и возврата. Приложение автоматизирует процесс покупки билетов как в онлайн-режиме, так и через кассу. Кроме того, система предоставляет пользователям возможность зарегистрироваться и просмотреть доступные рейсы. Приложение включает механизм взаимодействия с банковской системой для обработки оплаты и возврата средств.

Мы разобрались с DOM API, поняли связь между исполняемым скриптом и DOM-деревом. Отобразили данные с помощью JavaScript и DOM API.

Задание

1. Реализовать функции для отображения, добавления, редактирования и удаления информации на HTML-странице.
2. Использовать предоставленные фрагменты кода для демонстрации функциональности.

Выполненные шаги:

1. Загрузка скрипта

Для правильной последовательности загрузки был добавлен атрибут defer в тег <script>:

<script src="script.js" defer></script>

Это обеспечивает выполнение скрипта только после полной загрузки HTML-документа.

2. Навигация между страницами

Функция navigateTo реализует переход между страницами путем управления классами активности (active) у элементов форм:

function navigateTo(formId) {

document.querySelectorAll('.form').forEach(form => {

form.classList.remove('active');

});

```
const home = document.getElementById('home-page');
        if (formId === 'home-page') {
                home.classList.add('active');
                return;
        }
        if (formId === 'browsing') {
                const targetForm = document.getElementById(formId);
                targetForm.classList.add('active');
                findTickets();
                currentPage = 0;
                return;
        }
const targetForm = document.getElementById(formId);
if (targetForm) {
                home.classList.add('active');
targetForm.classList.add('active');
} else {
console.error(`Form with id "${formId}" not found.`);
        }
}
```

Эта функция проверяет наличие формы с указанным идентификатором и активирует её, одновременно сбрасывая состояние других форм.

3. Пагинация

Добавлена возможность отображать найденные билеты по 3 на странице и переключать страницы с помощью функций renderTickets, next_page и prev_page:

```
function prev_page() {
	currentPage = currentPage - 1;
	new_start = currentPage * 3;
	if (new_start < 0) {
		currentPage = Math.floor((filteredTickets.length - 1) / 3);
		new_start = currentPage * 3;
		renderTickets(filteredTickets.slice(new_start, Math.min(new_start
+ 3, filteredTickets.length)));
	} else {
		renderTickets(filteredTickets.slice(new_start, Math.min(new_start
+ 3, filteredTickets.length)));
	}
}
function next_page() {
	currentPage = currentPage + 1;
	new_start = currentPage * 3;
	if (new_start >= filteredTickets.length) {
		renderTickets(filteredTickets.slice(0, Math.min(3,
filteredTickets.length)));
		currentPage = 0;
	} else {
		renderTickets(filteredTickets.slice(new_start, Math.min(new_start
+ 3, filteredTickets.length)));
	}
}


function renderTickets(tickets) {

	const ticketsContainer = document.getElementById('ticket_container');

ticketsContainer.innerHTML = '';

const ticketsToRender = tickets.slice(0, 3);

ticketsToRender.forEach(ticket => {

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

ticketElement.innerHTML = ticketTemplate;

ticketElement.querySelector('.start-place').textContent =
townManager.getObjByValue(ticket.Obj.start_place).Obj.text;

ticketElement.querySelector('.end-place').textContent =
townManager.getObjByValue(ticket.Obj.end_place).Obj.text;
```

```
ticketElement.querySelector('.type-field').textContent =
ticketTypeManager.getObjByValue(ticket.Obj.type).Obj.text;

ticketElement.querySelector('.start-date').textContent =
ticket.Obj.start_date.toLocaleDateString(

        'ru-RU', {

                day: '2-digit',

                month: '2-digit',

                year: 'numeric',

                hour: '2-digit',

                minute: '2-digit',

        });

ticketElement.querySelector('.end-date').textContent =
ticket.Obj.finish_date.toLocaleDateString(

        'ru-RU', {

                day: '2-digit',

                month: '2-digit',

                year: 'numeric',

                hour: '2-digit',

                minute: '2-digit',

        });

ticketElement.querySelector('.cost').textContent = ticket.Obj.cost + ' ₽';

ticketsContainer.appendChild(ticketElement);

});

}
```

Функции работают с глобальными переменными

```
let currentPage;
```

```
let filteredTickets;
```

4. Фильтрация билетов

Функция findTickets позволяет фильтровать билеты по начальной и конечной точке маршрута:

```
function findTickets() {
    const ticket_properties = {};
    const startPlace = document.getElementById('start-point').value;
    if (startPlace !== 'none') {
        ticket_properties["start_place"] = startPlace;
    }
    const endPlace = document.getElementById('finish-point').value;
    if (endPlace !== 'none') {
        ticket_properties["end_place"] = endPlace;
    }
    const type = document.getElementById('type').value;
    if (type !== 'none') {
        ticket_properties["type"] = type;
    }
    const startDate = document.getElementById('input_date_start').value;
    if (startDate !== '') {
        ticket_properties["start_date"] = new Date(startDate);
    }
    const endDate = document.getElementById('input_date_finish').value;
    if (endDate !== '') {
        ticket_properties["finish_date"] = new Date(endDate);
```

```
        }
        const minCost = document.getElementById('input_price_min').value;
        if (minCost !== '') {
                ticket_properties["min_cost"] = minCost;
        }
        const maxCost = document.getElementById('input_price_max').value;
        if (maxCost !== '') {
                ticket_properties["max_cost"] = maxCost;
        }
        ticket_properties['purchased'] = false;
        filteredTickets = ticketManager.getObjs(0, 0, ticket_properties);
        console.log(filteredTickets);
        renderTickets(filteredTickets.slice(0, Math.min(3, filteredTickets.length)));
}
```

Она принимает значения из элементов select и на основе них фильтрует массив tickets.

5. Шаблон отображения билетов

Для отображения билетов используется HTML-шаблон:

```
const ticketTemplate = `
    <div class="ticket_background">
        <div class="info_vb">
            <div class="info_container">
                <div class="ticket_field">
                    <label for="start-place">Начало:</label>
                    <span class="start-place"></span>
```

```
    </div>
    <div class="ticket_field">
        <label for="end-place">Конец:</label>
        <span class="end-place"></span>
    </div>
    <div class="ticket_field">
        <label for="type">Вид билета:</label>
        <span class="type-field"></span>
    </div>
</div>
<div class="info_container">
    <div class="ticket_field">
        <label for="start-date">Отправление:</label>
        <span class="start-date"></span>
    </div>
    <div class="ticket_field">
        <label for="end-date">Прибытие:</label>
        <span class="end-date"></span>
    </div>
    <div class="ticket_field">
        <label for="cost">Стоимость:</label>
        <span class="cost"></span>
    </div>
</div>
```

```
        </div>
    </div>`;
```

Этот шаблон наполняется данными из отфильтрованного массива и добавляется в DOM.

6. Обновления стилей

Также были внесены соответствующие изменения в файл styles.css.

Код выполненного задания находится в репозитории GitHub по ссылке https://github.com/ScherbachenyaMIK/web-site_for_rw

Выводы

- Реализованные функции позволили управлять DOM-элементами динамически и эффективно.
- Использование предоставленного кода помогло освоить базовые принципы работы с DOM API.
- Фильтрация данных и пагинация улучшили пользовательский интерфейс, обеспечив удобство взаимодействия с системой.
- Все задачи выполнены в соответствии с требованиями задания.