**МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет Математики и Информатики**

**Департамент Информатики**

JAVA SCRIPT

Индивидуальная работа № 3

Проверил: Нартя Никита

Выполнил: Главчева Марина, IA2303

Кишинев, 2024

# Теоретическая часть

Основой HTML-документа являются теги.

В соответствии с объектной моделью документа («Document Object Model», коротко DOM), каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом.

Все эти объекты доступны при помощи JavaScript, мы можем использовать их для изменения страницы.

Например, document.body – объект для тега <body>.

# Задача

### Настройка и структурирование проекта

1. Создайте файл index.html
2. Создайте файл script.js
3. Создайте файл style.css
4. Подключите файл со стилями и js-кодом к HTML-странице.

### Представление транзакции

1. Создайте массив transactions, который будет содержать объекты транзакций.
2. Каждый объект транзакции должен иметь следующие поля:
   1. id: уникальный идентификатор транзакции.
   2. date: дата и время добавления транзакции.
   3. amount: сумма транзакции.
   4. category: категория транзакции.
   5. description: описание транзакции.

### Отображение транзакций

1. Создайте пустую таблицу, куда в дальнейшем Вы будете добавлять транзакции.
2. Таблица должна содержать следующие столбцы:
   1. ID
   2. Дата и Время
   3. Категория транзакции
   4. Краткое описание транзакции
   5. Действие (кнопка удаления транзакции)

### Добавление транзакций

1. Создайте функцию addTransaction(),
2. В функции addTransaction():
   1. Создайте объект транзакции с данными из формы.
   2. Добавьте созданный объект в массив transactions.
   3. Создайте новую строку таблицы с данными из объекта транзакции и добавьте её в таблицу.
      1. Если транзакция совершена на положительную сумму, то строка таблицы должна быть зеленым цветом, иначе красным.
      2. В колонке description отображайте краткое описание транзакции (первые 4 слова).

### Управление транзакциями

1. В каждой строке таблицы добавьте кнопку удаления.
2. При клике на кнопку удаления получите идентификатор транзакции и удалите соответствующую строку таблицы и удалите данную транзакцию из массива.
3. Обработчик событий на клик на кнопку определите для элемента <table>

### Подсчет суммы транзакции

1. Напишите функцию calculateTotal(), которая будет вызываться после добавления или удаления транзакции.
2. Отобразите общую сумму на странице, например, в отдельном элементе.

### Отображение полное транзакции

1. В файле index.html создайте блок для отображения подробного описания транзакции.
2. При нажатии на строку с транзакцией в таблице, отображайте полное описание в указанном месте.

### Добавление транзакции

1. Добавьте форму на страницу для добавления транзакции в таблицу (для категории используйте select).
2. Доп. задание: валидируйте форму на наличие ошибок.

# Практическая часть

Файл index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Transaction Management</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <h1>Transaction Management</h1>

    <form id="transaction-form">

        <label for="date">Date:</label>

        <input type="datetime-local" id="date" required>

        <label for="amount">Amount:</label>

        <input type="number" id="amount" required>

        <label for="category">Category:</label>

        <select id="category" required>

            <option value="Income">Income</option>

            <option value="Expense">Expense</option>

        </select>

        <label for="description">Description:</label>

        <input type="text" id="description" required>

        <button type="submit">Add Transaction</button>

    </form>

    <table id="transaction-table">

        <thead>

            <tr>

                <th>ID</th>

                <th>Date and Time</th>

                <th>Category</th>

                <th>Description</th>

                <th>Action</th>

            </tr>

        </thead>

        <tbody>

        </tbody>

    </table>

    <div id="transaction-details">

        <h2>Transaction Details</h2>

        <p id="full-description"></p>

    </div>

    <p id="total-amount">Total Amount: 0</p>

    <script src="script.js"></script>

</body>

</html>

Файл script.js

let transactions = [];

let nextId = 1;

document.getElementById('transaction-form').addEventListener('submit', function(event) {

    event.preventDefault();

    addTransaction();

});

function addTransaction() {

    const date = document.getElementById('date').value;

    const amount = parseFloat(document.getElementById('amount').value);

    const category = document.getElementById('category').value;

    const description = document.getElementById('description').value;

    const transaction = {

        id: nextId++,

        date: new Date(date).toLocaleString(),

        amount: amount,

        category: category,

        description: description

    };

    transactions.push(transaction);

    appendTransactionToTable(transaction);

    calculateTotal();

}

function appendTransactionToTable(transaction) {

    const table = document.getElementById('transaction-table').getElementsByTagName('tbody')[0];

    const row = table.insertRow();

    row.setAttribute('data-id', transaction.id);

    const idCell = row.insertCell(0);

    const dateCell = row.insertCell(1);

    const categoryCell = row.insertCell(2);

    const descriptionCell = row.insertCell(3);

    const actionCell = row.insertCell(4);

    idCell.textContent = transaction.id;

    dateCell.textContent = transaction.date;

    categoryCell.textContent = transaction.category;

    descriptionCell.textContent = transaction.description.split(' ').slice(0, 4).join(' ') + '...';

    actionCell.innerHTML = '<button onclick="deleteTransaction(' + transaction.id + ')">Delete</button>';

    if (transaction.amount > 0) {

        row.classList.add('green');

    } else {

        row.classList.add('red');

    }

    row.addEventListener('click', function() {

        showTransactionDetails(transaction);

    });

}

function deleteTransaction(id) {

    transactions = transactions.filter(transaction => transaction.id !== id);

    const row = document.querySelector(`tr[data-id='${id}']`);

    row.remove();

    calculateTotal();

}

function calculateTotal() {

    const total = transactions.reduce((acc, transaction) => acc + transaction.amount, 0);

    document.getElementById('total-amount').textContent = 'Total Amount: ' + total;

}

function showTransactionDetails(transaction) {

    document.getElementById('full-description').textContent = transaction.description;

}

Файл style.css

body {

    font-family: Arial, sans-serif;

    margin: 20px;

}

form {

    margin-bottom: 20px;

}

table {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 20px;

}

table, th, td {

    border: 1px solid black;

}

th, td {

    padding: 10px;

    text-align: left;

}

tbody tr.green {

    background-color: #d4edda;

}

tbody tr.red {

    background-color: #f8d7da;

}

#transaction-details {

    margin-top: 20px;

}

#transaction-details p {

    margin: 0;

}

#total-amount {

    font-weight: bold;

}

# Краткое описание особенностей реализации:

Структура данных:

* Массив транзакций: Используется массив transactions для хранения всех транзакций.
* Уникальные идентификаторы: Для каждой транзакции генерируется уникальный идентификатор с помощью переменной nextId.

Обработка событий:

* Отправка формы: Обработчик события submit для формы транзакций предотвращает стандартное поведение и вызывает функцию добавления транзакции.

Создание и добавление транзакции:

* Считывание данных из формы: Значения из формы считываются и преобразуются (например, дата и сумма).
* Создание объекта транзакции: Каждый объект транзакции содержит id, date, amount, category и description.
* Добавление в массив: Объект транзакции добавляется в массив transactions.
* Отображение в таблице: Функция appendTransactionToTable добавляет строку в таблицу для отображения новой транзакции.

Отображение данных:

* Создание строки таблицы: Каждая строка таблицы включает ячейки для всех атрибутов транзакции и кнопку удаления.
* Цветовая индикация: Строки таблицы окрашиваются в зеленый цвет для положительных сумм и в красный цвет для отрицательных.
* Краткое описание: В колонке описания отображаются первые четыре слова описания транзакции.

Удаление транзакции:

* Удаление из массива и таблицы: При удалении транзакции строка удаляется из таблицы, а соответствующий объект из массива transactions.
* Обновление общей суммы: После удаления транзакции пересчитывается и отображается общая сумма.

Подсчет общей суммы:

* Функция calculateTotal: Суммирует значения amount всех транзакций и обновляет элемент с ID total-amount.

# Ответы на контрольные вопросы

1. Каким образом можно получить доступ к элементу на веб-странице с помощью JavaScript?

Основные методы, которые можно использовать для этого, включают:

getElementById: Этот метод используется для получения элемента по его ID. Он возвращает единственный элемент, соответствующий указанному ID.

querySelector и querySelectorAll: Эти методы используются для получения элемента или элементов, соответствующих указанному CSS-селектору.

1. Что такое делегирование событий и как оно используется для эффективного управления событиями на элементах DOM?

Делегирование событий — это техника, при которой обработчик события устанавливается на родительском элементе, а не на каждом отдельном элементе. Это позволяет управлять событиями на динамически добавленных элементах и снижает количество обработчиков, улучшая производительность.

Преимущества

* Снижение нагрузки на память: Один обработчик для множества элементов.
* Поддержка динамически добавленных элементов: Работает с элементами, добавленными после установки обработчика.
* Упрощение кода: Централизованное управление обработчиками событий.

1. Как можно изменить содержимое элемента DOM с помощью JavaScript после его выборки?

Чтобы изменить содержимое элемента DOM с помощью JavaScript, можно использовать методы textContent, innerHTML или innerText.

textContent: Изменяет текстовое содержимое элемента, игнорируя HTML-разметку.

innerHTML: Изменяет HTML-содержимое элемента, позволяя включать HTML-теги.

innerText: Изменяет текстовое содержимое, учитывая стили CSS (например, скрытые элементы).

Для доступа к элементу используются методы getElementById, querySelector или querySelectorAll:

getElementById: Получает элемент по его уникальному ID.

querySelector: Получает первый элемент, соответствующий указанному CSS-селектору.

querySelectorAll: Получает все элементы, соответствующие указанному CSS-селектору.

1. Как можно добавить новый элемент в DOM дерево с помощью JavaScript?

Чтобы добавить новый элемент в DOM дерево с помощью JavaScript, следует выполнить следующие шаги:

1. Создание нового элемента: Используйте метод `document.createElement(tagName)`, где `tagName` это строка с названием тега нового элемента (например, `"div"`, `"p"`, `"span"`, и т.д.). Этот метод создает новый элемент, но не добавляет его в DOM.

2. Настройка нового элемента: При необходимости настройте созданный элемент, установив его атрибуты, свойства или текстовое содержимое с помощью JavaScript.

3. Добавление нового элемента в DOM:

- Вставка в конец родительского элемента: Используйте методы `parentElement.appendChild(newElement)` или `parentElement.append(newElement)`, чтобы добавить новый элемент в конец родительского элемента.

- Вставка в начало родительского элемента: Используйте метод `parentElement.prepend(newElement)`, чтобы добавить новый элемент в начало родительского элемента.

- Вставка перед существующим элементом: Используйте метод `referenceElement.parentElement.insertBefore(newElement, referenceElement)`, чтобы вставить новый элемент перед существующим элементом `referenceElement`.

4. Удаление или замена существующего элемента: При необходимости, перед добавлением нового элемента, можно удалить или заменить существующий элемент с помощью методов `parentElement.removeChild(oldElement)` или `parentElement.replaceChild(newElement, oldElement)`.

# Вывод

В результате выполнения лабораторной работы по управлению транзакциями я приобрел(а) следующие навыки:

- Создание основной структуры HTML страницы с использованием таблиц, форм и блоков.

- Применение CSS для стилизации страницы и улучшения её внешнего вида.

- Использование JavaScript для добавления динамической функциональности, такой как добавление, удаление и отображение информации о транзакциях.

- Работа с массивами и объектами в JavaScript для хранения данных и управления списком транзакций.

- Освоение принципов делегирования событий для эффективного управления динамическими элементами на странице.

**Ссылка на репозиторий Git:**

# Список использованных источников

<https://learn.javascript.ru/dom-nodes>

<https://github.com/MSU-Courses/javascript_typescript/blob/main/lab/LI3/JS03.md>