Молдавский Государственный Университет
Факультет Математики и Информатики
Департамент Информатики

Отчет по лабораторной работе по дисциплине „JavaScript”

Выполнила: студентка группы **IAFR2403 R**
Alexandra Ivarovscaia
Проверил преподаватель:
Nartea Nichita, lector univ.

Кишинев, 2025

Индивидуальная (лабораторная) работа №3

Цель: ознакомиться с основами взаимодействия JS с DOM-деревом на основе веб-приложения для учета личных финансов.

Условие: создание веб-приложения для учета личных финансов.

Структура проекта:

1. файл index.html, содержащий HTML код веб-страницы
2. файл style.css, содержащий стилистическое оформление веб-страницы
3. файл script.js, содержащий JS код для взаимодействия с веб-страницей

Выполнение работы:

Представление транзакции**:** массив transactions, который содержит объекты транзакций. Каждый объект транзакции содержит следующие поля:

- id: уникальный идентификатор транзакции
- date: дата и время добавления транзакции
- amount: сумма транзакции
- category: категория транзакции
- description: описание транзакции

```
JS script.js > ...
1   let transactions = [];
2   let idGenerator = 1;
3   let totalAmount = 0;
4
5   /**
6    * @description class-template for transaction object
7    * @constructor
8    * @param {number} id - transaction id
9    * @param {string} date - transaction date
10   * @param {number} amount - transaction amount
11   * @param {string} category - transaction category
12   * @param {string} description - transaction description
13   */
14  class Transaction {
15      constructor (id, date, amount, category, description) {
16          this.id = id;
17          this.date = date;
18          this.amount = amount;
19          this.category = category;
20          this.description = description;
21      }
22  }
```

Рисунок 1: класс-конструктор объекта транзакций

Отображение транзакций: пустая таблица, куда добавляются транзакции. Таблица содердит следующие столбцы:

- ID
- Дата и Время
- Категория транзакции
- Краткое описание транзакции
- Действие (кнопка удаления транзакции)

```
<body>
<h1>Transactions:</h1>
<div class="transactions_table">
<table id="transactions_table" style="border: 1px solid black; border-collapse: collapse; text-align: center; width: 100%;">
    <tr>
        <th style="border: 3px solid black; border-collapse: collapse; background-color: lightblue;">ID</th>
        <th style="border: 3px solid black; border-collapse: collapse; background-color: lightblue;">Date</th>
        <th style="border: 3px solid black; border-collapse: collapse; background-color: lightblue;">Category</th>
        <th style="border: 3px solid black; border-collapse: collapse; background-color: lightblue;">Description</th>
        <th style="border: 3px solid black; border-collapse: collapse; background-color: lightblue;">Action</th>
    </tr>
</table></div>
```

Рисунок 2: HTML таблица для добавления транзакций

Добавление транзакций: функция addTransaction(), в которой создаётся объект транзакции с данными из формы. Созданный объект добавляется в массив transactions. В таблице создаётся новая строка с данными из объекта транзакции. Если транзакция совершена на положительную сумму, то строка таблицы выделена зеленым цветом, иначе красным.

```
function addTransaction (amount, description, date, category) {
    const id = idGenerator;
    idGenerator = idGenerator + 1;

    const addedTransaction = new Transaction(id, date, amount, category, description);
    transactions.push(addedTransaction);


    const row = document.createElement("tr");
    const cellId = document.createElement("td");
    const cellDate = document.createElement("td");
    const cellCategory = document.createElement("td");
    const cellDescription = document.createElement("td");
    const cellAction = document.createElement("td");


    cellId.textContent = addedTransaction.id;
    row.appendChild(cellId);
    cellDate.textContent = addedTransaction.date;
    row.appendChild(cellDate);
    cellCategory.textContent = addedTransaction.category;
    row.appendChild(cellCategory);
    cellDescription.textContent = addedTransaction.description;
    row.appendChild(cellDescription);
    const deleteButton = document.createElement("button");
    deleteButton.textContent = "Delete";
    cellAction.appendChild(deleteButton);
    row.appendChild(cellAction);
```

Рисунок 3: функция добавления транзакции в массив и в таблицу

```
if (category === "Expenses") {
    row.style.backgroundColor = "red";
    row.style.border = 'solid 1px black';
} else {
    row.style.backgroundColor = "green";
    row.style.border = 'solid 1px black';
}
```

Рисунок 4: цвет строки таблицы в зависимости от категории транзакции

Управление транзакциями: в каждой строке таблицы добавлена кнопка удаления. При клике на кнопку удаляется соответствующая строка таблицы и данная транзакция из массива.

```
deleteButton.addEventListener("click", deleteTransaction);

document.getElementById("transactions_table").appendChild(row);
```

Рисунок 5: функция удаления транзакции из массива и таблицы

```
/**
   * @description delete transaction after Delete button is clicked function
   * deletes transaction object from array as well as entire transaction information row from HTML document
   * after Delete button is clicked all Transaction Details fields are cleared
   * recalculation on total amount included
   */
function deleteTransaction() {
    const row = this.closest("tr");
    const transactionID = row.querySelector("td").textContent;
    row.remove();
    transactions = transactions.filter(t => t.id !== Number(transactionID));
    document.getElementById("details_id").textContent = "";
    document.getElementById("details_date").textContent = "";
    document.getElementById("details_amount").textContent = "";
    document.getElementById("details_category").textContent = "";
    document.getElementById("details_description").textContent = "";
    calculateTotalAmount();
}
```

Рисунок 6: функция удаления транзакции из массива и таблицы (продолжение)

Подсчет суммы транзакции: функция calculateTotalAmount(), которая вызывается после добавления или удаления транзакции. Она отображает общую сумму на странице в отдельном элементе.

```
/**
   * @description calculate total trasaction amount method
   * and post the result in the HTML document
   */
function calculateTotalAmount() {
    totalAmount = transactions.reduce((sum, transaction) => sum + transaction.amount, 0);
    document.getElementById("totalAmount").textContent = totalAmount;
}

calculateTotalAmount();
```

Рисунок 7: функция подсчёта суммы транзакций

Отображение полной информации о транзакции: в файле index.html создан блок для отображения подробного описания последней добавленной транзакции. При удалении транзакции блок очищается от описания.

```
document.getElementById("details_id").textContent = addedTransaction.id;
document.getElementById("details_date").textContent = addedTransaction.date;
document.getElementById("details_amount").textContent = addedTransaction.amount;
document.getElementById("details_category").textContent = addedTransaction.category;
document.getElementById("details_description").textContent = addedTransaction.description;
```

Рисунок 8: отображение деталей транзакции в специальном блоке

Добавление транзакции через форму: на странице отображается форма для добавления транзакции в таблицу. Поля формы проверяются на наличие пустых значений или на соответствие нумерическому значению (в случае amount).

```
/**
   * @description function to wait for Submit button to be clicked
   * and get the appropriate values from the HTML document
   * validation for all fields to be populated included
   * validation for amount numeric value included
   * after Submit button is clicked all form fields are cleared
   */
document.getElementById("transaction_submit").addEventListener("click", function() {
    const amount = Number(document.getElementById("transaction_amount").value);
    const description = document.getElementById("transaction_description").value;
    const date = document.getElementById("transaction_date").value;
    const category = document.getElementById("transaction_category").value;

    if (date == "" || amount == "" || category == "" || description == "") {
        alert("All fields must be filled out!");
        return false;
      } else if (isNaN(amount)) {
        alert("Amount must be a numeric value!");
        return false;
      }


    addTransaction(amount, description, date, category);

    document.getElementById("transaction_amount").value = "";
    document.getElementById("transaction_description").value = "";
    document.getElementById("transaction_date").value = "";
    document.getElementById("transaction_category").value = "";


});
```

Рисунок 9: функция добавления транзакции через форму

Ссылка на GIT репозиторий: https://github.com/

Контрольные вопросы

1. Каким образом можно получить доступ к элементу на веб-странице с помощью JavaScript?
   ПВыбор элементов в основном выполняется с помощью этих методов: querySelector; querySelectorAll. Они позволяют выполнить поиск HTML-элементов по CSS-селектору. При этом querySelector выбирает один элемент, а querySelectorAll – все. Кроме них имеются ещё: getElementById; getElementsByClassName; getElementsByTagName; getElementsByName. Также в JavaScript имеются очень полезные методы: matches – позволяет проверить соответствует ли HTML-элемент указанному CSS-селектору; closest – позволяет найти для HTML-элемента его ближайшего предка, подходящего под указанный CSS-селектор (поиск начинается с самого элемента); contains – позволяет проверить содержит ли данный узел другой в качестве потомка (проверка начинается с самого этого узла).
2. Что такое делегирование событий и как оно используется для эффективного управления событиями на элементах DOM?
   Делегирование событий является полезным шаблоном, так как позволяет отслеживать события на множестве элементов с помощью только одного обработчика. Для работы делегирования событий нужно 3 шага: определить родителя элементов для отслеживания событий, прикрепить на него обработчик событий, использовать event.target для выбора целевого элемента. Делегирование событий - это мощный и эффективный способ обработки пользовательских событий в веб-разработке. Оно позволяет обрабатывать события наиболее эффективным способом, поскольку требует привязки к элементу только одного слушателя событий, а не нескольких.
3. Как можно изменить содержимое элемента DOM с помощью JavaScript после его выборки?
   Для создания новых элементов в DOM используется метод document.createElement(). Этот метод создает новый элемент, но не добавляет его в DOM. Чтобы добавить созданный элемент в DOM, используются методы appendChild() или insertBefore(). Метод appendChild() добавляет элемент в конец родительского элемента, а insertBefore() позволяет вставить элемент перед другим дочерним элементом. После добавления элемента в DOM, можно изменить его стили и атрибуты. Для изменения стилей используется свойство style, а для изменения атрибутов — методы setAttribute() и getAttribute(). Можно изменять цвета, размеры, шрифты и другие стили, чтобы создать привлекательный интерфейс. Также можно добавлять и изменять атрибуты, такие как идентификаторы и классы, чтобы управлять элементами с помощью CSS и JavaScript. Для удаления элементов из DOM используйте метод removeChild() или свойство remove().
4. Как можно добавить новый элемент в DOM дерево с помощью JavaScript?

Создание новых элементов — это первый шаг в процессе динамического изменения DOM. После создания элемента, можно настроить его свойства, добавить стили и атрибуты, а затем вставить его в нужное место в DOM-дереве. Это позволяет создавать сложные интерфейсы и динамически изменять их в зависимости от действий пользователя или других факторов. Для создания новых элементов в DOM используется метод document.createElement(). Этот метод создает новый элемент, но не добавляет его в DOM. Чтобы добавить созданный элемент в DOM, используйте методы appendChild() или insertBefore(). Метод appendChild() добавляет элемент в конец родительского элемента, а insertBefore() позволяет вставить элемент перед другим дочерним элементом.

Список литературы

1. MSU-Courses GitHub – курс JavaScript Государственного Университета Молдовы
2. W3Schools JavaScript Tutorial – базовый курс JavaScript
3. Freecodecamp – курс JavaScript для начинающих
4. «Программирование на JavaScript», Васильев А.Н. – учебное пособие для самостоятельного обучения