Отчет по Лабораторной работе №3

**«Взаимодействие JavaScript с DOM»**

**Цель работы:**  
Научиться работать с DOM-деревом в браузере: создавать, обновлять и удалять элементы, обрабатывать события и поддерживать актуальное состояние данных в интерфейсе на примере приложения для учёта личных финансов.

## 1. Структура проекта

* **index.html** — содержит разметку страницы: форму добавления транзакции, пустую таблицу для вывода и блоки для общей суммы и полного описания.
* **style.css** — стили для оформления формы, таблицы и элементов интерфейса.
* **script.js** — вся логика работы: управление массивом транзакций, добавление/удаление строк, подсчёт суммы и показ полного описания.

## 2. Представление транзакции

* Заведён массив transactions, элементы которого — объекты с полями:
  1. id — уникальный идентификатор,
  2. date — строка с датой и временем (toLocaleString()),
  3. amount — число (сумма),
  4. category — строка,
  5. description — полное описание.

## 3. Отображение транзакций

* В HTML создана пустая таблица <table id="transaction-table"> с колонками:
  + **ID**
  + **Дата и Время**
  + **Категория**
  + **Краткое описание** (первые 4 слова)
  + **Действие** (кнопка удаления)

## 4. Добавление транзакций

* Функция addTransaction() вызывается при отправке формы.
* Она читает значения из полей, валидирует их, создаёт объект транзакции и добавляет его в массив.
* Затем динамически создаётся новая строка таблицы с данными из объекта.
* Для положительной суммы строка помечается классом .positive (зелёный), для отрицательной — .negative (красный).

## 5. Удаление транзакций

* В каждой строке есть кнопка «Delete».
* Обработчик клика назначен на весь <table> (делегирование).
* При нажатии на кнопку строка и соответствующий объект удаляются из DOM и из массива.

## 6. Подсчёт общей суммы

* Функция calculateTotal() перебирает массив через reduce() и выводит результат в элемент <div id="total">.
* Вызывается после каждого добавления и удаления транзакции.

## 7. Отображение полного описания

* Под таблицей есть блок <div id="description">.
* При клике на строку <tr> (делегирование на <tbody>) скрипт находит соответствующий объект и показывает его полное описание в этом блоке.

## 8. Валидация формы

* Прежде чем создать транзакцию, скрипт проверяет, что сумма — число, категория и описание не пустые.
* В случае ошибки выводится alert() и транзакция не добавляется.

## 9. Документирование кода

* Каждая функция сопровождается JSDoc-комментарием, описывающим её назначение, параметры и возвращаемые значения.

## 10. Ответы на вопросы

**1. Как получить доступ к элементу на веб-странице?**  
Можно обратиться к элементу несколькими способами. Если у него есть уникальный id, самый простой путь — вызвать document.getElementById с этим идентификатором. Для более гибкого поиска подойдут CSS-селекторы: document.querySelector вернет первый совпавший элемент, а document.querySelectorAll — список всех подходящих узлов. Есть также старые методы getElementsByClassName и getElementsByTagName, но их обычно заменяют на querySelector(All) из-за единообразия.

**2. Что такое делегирование событий и зачем оно нужно?**  
Делегирование — это приём, при котором один обработчик события навешивается не на каждый дочерний элемент, а на их общего родителя. При клике на любой вложенный элемент событие «всплывает» вверх через дерево DOM, и родительский обработчик перехватывает его. Это позволяет использовать всего один слушатель вместо множества, автоматически обрабатывать динамически добавленные элементы и экономить ресурсы.

**3. Как изменить содержимое элемента после его выбора?**  
После того как нужный узел получен, его текст можно заменить свойством textContent, которое обновит только текст без разбора HTML. Если нужно вставить HTML-разметку, применяют innerHTML. Для изменения атрибутов (например, ссылки или изображениия) применяют соответствующие свойства элемента, например href или src. Также можно управлять классами через classList для изменения стилей.

**4. Как добавить новый элемент в DOM-дерево?**  
Сначала создают узел командой createElement, затем заполняют его содержимым и настраивают нужные атрибуты или классы. После этого этот новый элемент вставляют в документ, вызвав у родительского узла метод appendChild (добавить в конец) или insertBefore (вставить перед указанным элементом). Такой подход позволяет динамически изменять интерфейс без перезагрузки страницы.