// - Объявление переменных, с которыми будем работать

const todoInput = document.querySelector('.todo-input');

const todoButton = document.querySelector('.todo-button');

const todoList = document.querySelector('.todo-list');

const filterOption = document.querySelector('#filter-todo');

// - Вешаем прослушку событий на элементы

todoButton.addEventListener('click', addToDo);

todoList.addEventListener('click', deleteCheck);

filterOption.addEventListener('click', filterToDo);

/\*\*

 \* - Либо так:

 \*& todoForm.addEventListener('submit', addToDo);

 \*/

function addToDo*(event)* {

    event.preventDefault(); // - Убираем стандартное поведение form

    // - Создаем контейнер для задачи

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

    todoDiv.classList.add('todo');

    // - Создаем саму задачу

    const newTodo = document.createElement('li');

    newTodo.innerText = todoInput.value;

    newTodo.classList.add('todo-item');

    todoDiv.append(newTodo);

    // - Обнуляем input и возвращаем на него фокус

    todoInput.value = '';

    todoInput.focus();

    // - Создаем кнопку выполнения задачи

    const completedButton = document.createElement('button');

    completedButton.innerHTML = '<i class="fa-solid fa-check"></i>';

    completedButton.classList.add('complete-btn');

    todoDiv.append(completedButton);

    // - Создаем кнопку удаления задачи

    const trashButton = document.createElement('button');

    trashButton.innerHTML = '<i class="fa-solid fa-trash"></i>'; // - Устанавливаем иконку на кнопку

    trashButton.classList.add('trash-btn'); // - Добавляем класс для button

    todoDiv.append(trashButton);

    // - Выводим в window

    todoList.append(todoDiv);

}

// - Создание функции по удалению задачи и её выполнению

function deleteCheck*(e)* {

    const item = e.target; // - Свойство target является ссылкой на объект, который был инициатором события

    if (item.classList[0] === 'trash-btn') {

        const todo = item.parentElement; // - Родительский элемент - div с классом todo (создается на строке 17-19)

        todo.classList.add('fall');

        todo.addEventListener('transitionend', function*()* { // - Событие transitionend срабатывает, когда CSS transition закончил своё выполнение. То есть срабатывает, когда анимация завершена

            todo.remove();

        })

    } else if (item.classList[0] === 'complete-btn') {

        const todo = item.parentElement; // - Родительский элемент - div с классом todo (создается на строке 17-19)

        todo.classList.toggle('completed');

    }

 }

 /\*\*

  \* Свойство classList вовращает псевдомассив, который содержит все классы элемента.

  \* Допустим у элемента есть два класса: 'class\_1' и 'class\_2'.

  \* class\_1 имеет индекс 0, class\_2 - индекс 1.

  \* Чтобы получить доступ к одному из этих классов, можно воспользоваться свойством classList: element.classList[...].

  \* element.classList[0] вернет 'class\_1', element.classList[1] вернет 'class\_2'

  \*/

 function filterToDo*(e)* {

     const todos = todoList.childNodes;

     /\*\*

      \* Аттрибут childNodes возвращает коллекцию дочерних элементов данного элемента.

      \* В данном случае переменной todos присваивается коллекция дочерних элементов элемента todoList

      \*/

     todos.forEach(function*(todo)* {

         switch (e.target.value) {

            case 'All':

                todo.style.display = 'flex';

                break;

            case 'Completed':

                if (todo.classList.contains('completed')) {

                    todo.style.display = 'flex';

                } else {

                    todo.style.display = 'none';

                }

                break;

            case 'Uncompleted':

                if (!todo.classList.contains('completed')) {

                    todo.style.display = 'flex';

                } else {

                    todo.style.display = 'none';

                }

                break;

         }

     })

 }