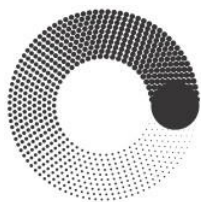


федеральное государственное автономное образовательное  
учреждение высшего образования



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
(ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ)  
(Факультет информационных технологий)

*(Институт Принтмедиа и информационных технологий) Кафедра  
Информатики и информационных технологий*

направление подготовки 09.03.02  
«Информационные системы и технологии»

## ЛАБОРАТОРНАЯ РАБОТА № 2

Дисциплина: Функциональное программирование.

Тема: Применение функционального программирования в JS.

Выполнил(а): студент(ка) группы 221-3711

Мироненко Р. Е.

(Фамилия И.О.)

Дата, подпись 18.02.2025

(Дата) (Подпись)

Проверил: \_\_\_\_\_

(Фамилия И.О., степень, звание)

\_\_\_\_\_  
(Оценка)

Дата, подпись \_\_\_\_\_

(Дата)

(Подпись)

Замечания:

---

---

---

## **Москва 2025**

**Цель:** Применить принципы функционального программирования для разработки небольшого веб-приложения.

### **Задание:**

Разработайте веб-приложение "Список задач", которое позволяет пользователю:

- Добавлять новые задачи.
- Отмечать задачи как выполненные.
- Удалять задачи.
- Фильтровать задачи по статусу (выполненные/невыполненные).

### **Требования:**

- Используйте принципы функционального программирования, такие как иммутабельность данных и чистые функции.
- Используйте функции высшего порядка для обработки списка задач.
- Веб-приложение должно быть реализовано с использованием HTML, CSS и JavaScript.
- Интерфейс должен быть интуитивно понятным и удобным для пользователя.

## Ход работы:

Гит: <https://github.com/Roman784/FuncProg.git>

Листинг 1 index.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Список задач</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1 class="title">Список задач</h1>
  <input type="text" class="taskInput" id="taskInput" placeholder="Введите новую задачу" />
  <button class="addTaskButton" id="addTaskButton">Добавить задачу</button>

  <h2 class="title">Фильтр по статусу</h2>
  <select class="statusFilter" id="statusFilter">
    <option value="all">Все</option>
    <option value="completed">Выполненные</option>
    <option value="notCompleted">Невыполненные</option>
  </select>

  <ul class="taskList" id="taskList"></ul>

  <script src="script.js"></script>
</body>
</html>
```

Листинг 2 styles.css

```
body {
  font-family: Arial, sans-serif;
  margin: 20px;
  background-color: #f4f4f4;
}

.title {
  color: #333333;
}

.taskInput {
  border: 1px solid #e7e7e7;
  border-radius: 10px 0px 0px 10px;
  padding: 15px;
  padding-right: 70%;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}
```

```
    font-size: 16px;
}

.addTaskButton {
    background-color: #3498db;
    color: #ffffff;
    border: none;
    border-radius: 0px 10px 10px 0px;
    padding: 15px;
    cursor: pointer;
    font-size: 16px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.4);
}

.addTaskButton:hover {
    background-color: rgb(33, 113, 204);
    box-shadow: none;
}

.taskList {
    list-style-type: none;
    padding: 0;
}

.task {
    display: flex;
    align-items: center;
    justify-content: space-between;
    background-color: #ffffff;
    border-radius: 10px;
    padding: 15px;
    margin: 10px 0;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    font-size: 16px;
}

.completedCheckbox {
    width: 20px;
    height: 20px;
    color: #3498db;
}

.completedTask {
    color: rgb(129, 129, 129);
}

.deleteTaskButton {
    background-color: #e7412e;
    color: #ffffff;
    border: none;
```

```

border-radius: 10px;
padding: 10px 15px;
cursor: pointer;
font-size: 16px;
}

.deleteTaskButton:hover {
  background-color: #ad3224;
}

.statusFilter {
  padding: 10px 10px;
  border: 1px solid #3498db;
  border-radius: 10px;
  background-color: #ffffff;
  color: #333333;
  cursor: pointer;
  font-size: 16px;
}

```

Листинг 3 script.js

```

const taskInput = document.getElementById('taskInput');
const addTaskButton = document.getElementById('addTaskButton');
const statusFilter = document.getElementById('statusFilter');
const taskList = document.getElementById('taskList');

addTaskButton.addEventListener('click', addTask);
statusFilter.addEventListener('change', renderTasks);

window.onload = renderTasks();

function loadTasks() {
  const json = localStorage.getItem('tasks');
  return JSON.parse(json) || [];
}

function saveTasks(tasks) {
  const json = JSON.stringify(tasks);
  localStorage.setItem('tasks', json);
}

function renderTasks() {
  const tasks = loadTasks();
  const status = statusFilter.value;

  taskList.innerHTML = '';

  tasks.forEach((task, index) => {

```

```

        if (status === 'completed' && !task.completed) return;
        if (status === 'notCompleted' && task.completed) return;

        renderTask(task, index);
    });
}

function renderTask(task, index) {
    const taskItem = document.createElement('li');
    taskItem.className = 'task';
    taskItem.innerHTML = `
        <input type="checkbox" class="completedCheckbox" ${task.completed ? 'checked' : ''}
onchange="toggleTask(${index})">
        <span class="${task.completed ? 'completedTask' : ''}">${task.text}</span>
        <button class="deleteTaskButton" onclick="deleteTask(${index})">Удалить</button>
    `;
    taskList.appendChild(taskItem);
}

function addTask() {
    const tasks = loadTasks();
    const taskText = taskInput.value.trim();

    if (!taskText) return;

    const task = { text: taskText, completed: false };
    tasks.push(task);
    saveTasks(tasks);

    taskInput.value = '';
    renderTasks();
}

function toggleTask(index) {
    const tasks = loadTasks();
    const task = tasks[index];

    task.completed = !task.completed;
    saveTasks(tasks);

    renderTasks();
}

function deleteTask(index) {
    const tasks = loadTasks();
    tasks.splice(index, 1);
    saveTasks(tasks);

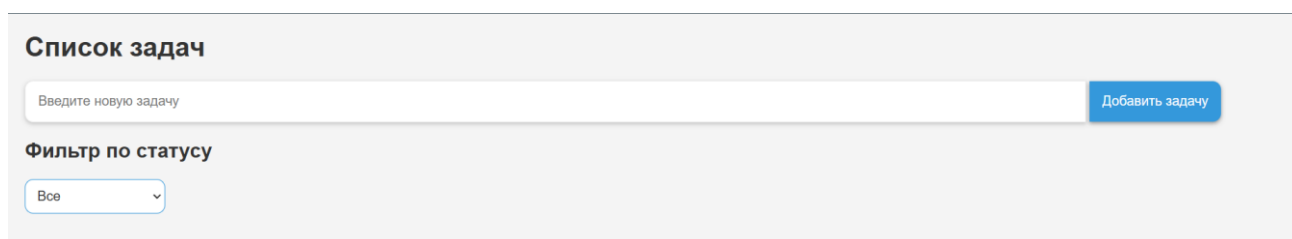
    renderTasks();
}

```

Методы `loadTasks` и `saveTasks` загружают и сохраняют данные в локальном хранилище в формате `json` соответственно.

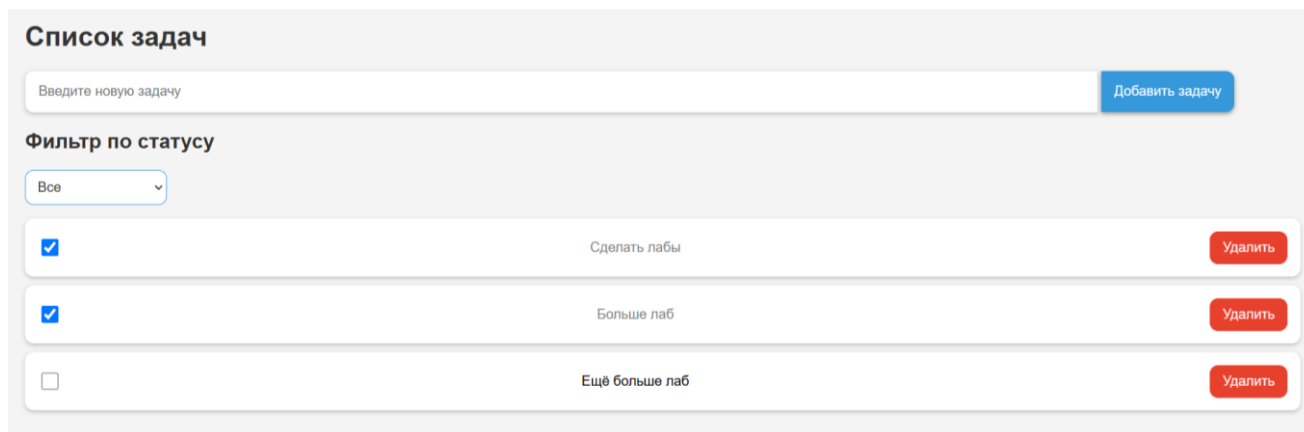
При загрузке страницы сразу вызывается функция `renderTasks`. Она загружает задачи из хранилища, проходит по ним `forEach`-ем и вызывает функцию `renderTask`. Она создаёт и настраивает новый `html` элемент с тегом `li`, этот элемент является плашкой в списке задач.

Функция `addTask` считывает значение из поля ввода и, если там что-то написано, создаёт новую задачу и добавляет к списку других. `toggleTask` получает индекс задачи и меняет её статус. `deleteTask` удаляет задачу. Все изменения сохраняются в хранилище.



The screenshot shows the main page of the application. At the top, there is a section titled "Список задач" (Task List). Below the title, there is a text input field with the placeholder "Введите новую задачу" (Enter a new task) and a blue button labeled "Добавить задачу" (Add task). Below the input field, there is a section titled "Фильтр по статусу" (Filter by status) with a dropdown menu currently set to "Все" (All). The task list itself is empty.

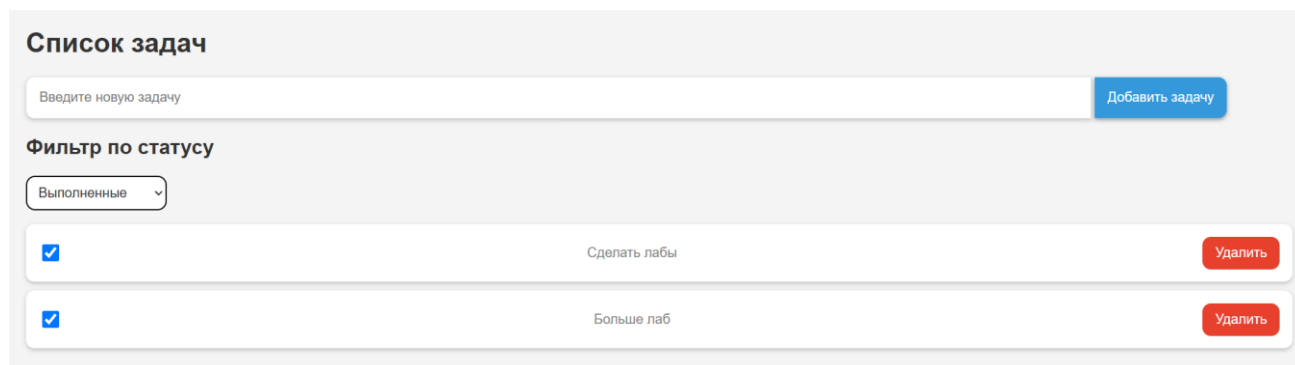
Рисунок 0.1 Главная страница.



The screenshot shows the task list interface with three tasks. The "Фильтр по статусу" (Filter by status) dropdown is still set to "Все" (All). The tasks are:

Checkbox	Task Text	Action
<input checked="" type="checkbox"/>	Сделать лабы	Удалить
<input checked="" type="checkbox"/>	Больше лаб	Удалить
<input type="checkbox"/>	Ещё больше лаб	Удалить

Рисунок 0.2 Создание задач.



The screenshot shows the task list interface with the "Фильтр по статусу" (Filter by status) dropdown set to "Выполненные" (Completed). Only the first two tasks from the previous screenshot are visible, both with checked checkboxes:

Checkbox	Task Text	Action
<input checked="" type="checkbox"/>	Сделать лабы	Удалить
<input checked="" type="checkbox"/>	Больше лаб	Удалить

Рисунок 0.3 Выполненные задачи.

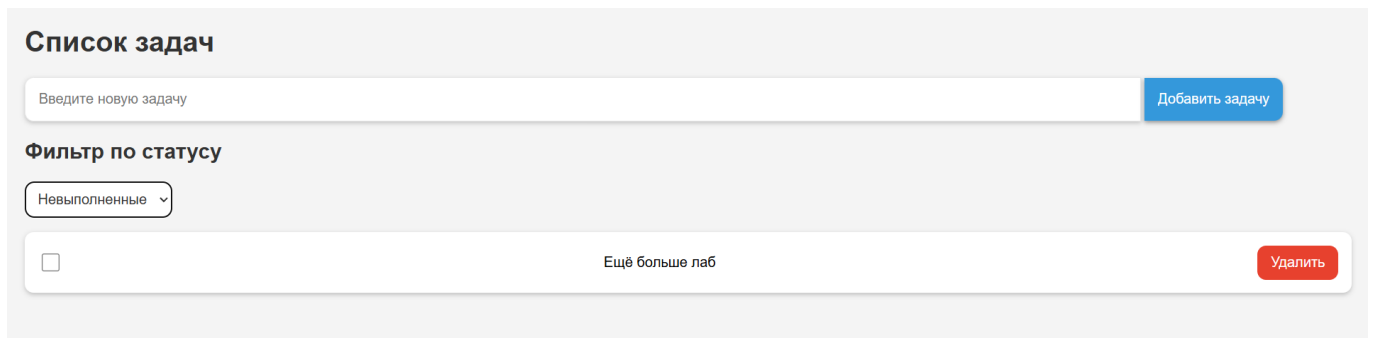


Рисунок 0.4 Невыполненные задачи.

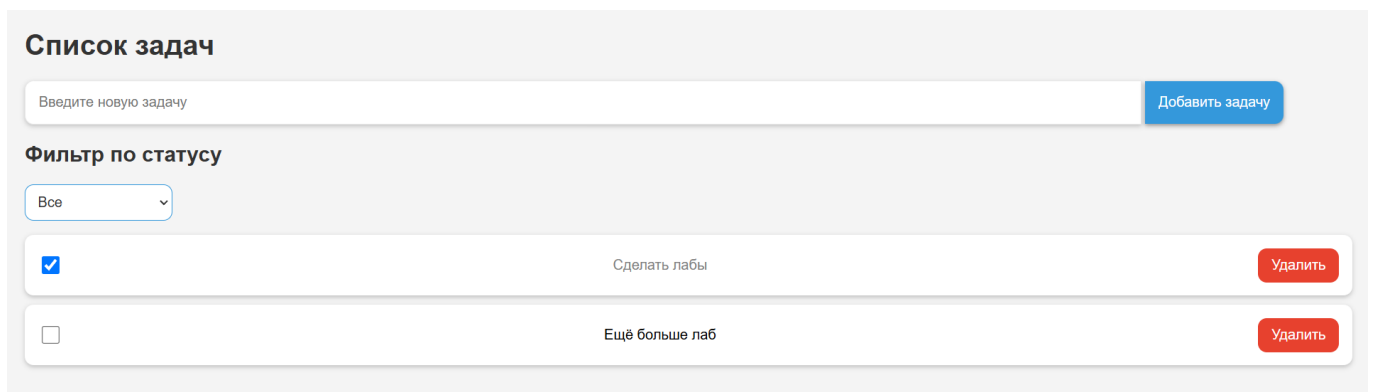


Рисунок 0.5 Удаление второй задачи.

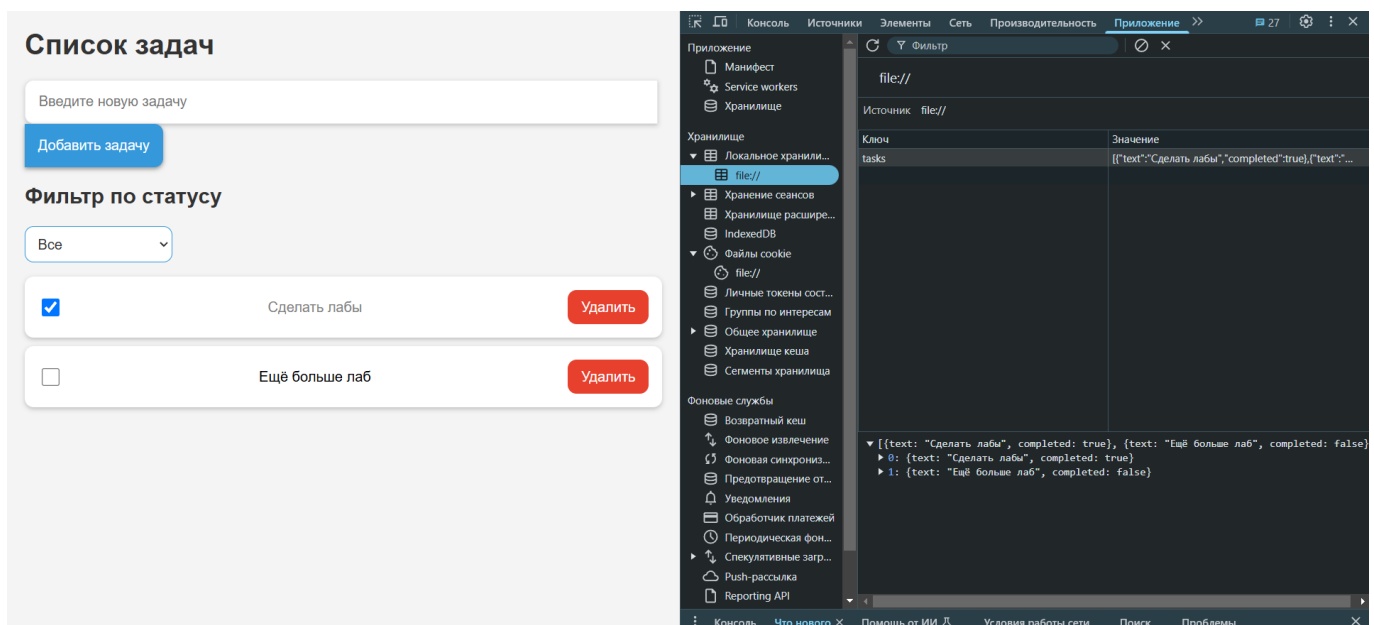


Рисунок 0.6 Локальное хранилище.