МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

УТВЕРЖДАЮ
Зав.кафедрой,
доцент, к. ф.-м. н.
_____ С. В. Миронов

ОТЧЕТ О ПРАКТИКЕ

студентки 3 курса 311 группы факультета компьютерных наук и информационных технологий
Никитенко Яны Валерьевны

вид практики: учебная
кафедра: информатики и программирования
курс: 3
семестр: 1
продолжительность: 2 нед., с 01.09.2025 г. по 30.12.2025 г.

Руководитель практики от университета,

доцент, к. ф.-м. н. _____ А. С. Иванова

Руководитель практики от организации (учреждения, предприятия),

доцент, к. ф.-м. н. _____ А. С. Иванова

Тема практики:«Проектная практика»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данный проект представляет собой одностраничное веб-приложение для управления списком дел "Сервис To-do разработанное в рамках практического задания по фронтенд-разработке [1]. Пользователь сможет просматривать, добавлять, удалять, копировать и редактировать задачи. А все данные будут сохраняться в локальное хранилище, чтобы они не терялись при перезагрузке страницы.

1 Отображение текущих задач

При загрузке страницы пользователь должен видеть список текущих задач(см. рисунок 1). Для этого реализовано две функции [2]:

1. Функция loadInitialTasks

   Она отвечает за получение списка задач. служит для загрузки начального списка задач из локального хранилища браузера (localStorage) или использования дефолтного массива [3] items, если сохранённых задач нет.

```
function loadInitialTasks() {
    const savedTasks = localStorage.getItem('items');
    return savedTasks ? JSON.parse(savedTasks) : [...items];
}
```

   Если savedTasks существует (не null/undefined), парсит JSON-строку в массив и возвращает его.

   Если сохранённых задач нет, возвращает копию исходного массива items (через spread-оператор [...items])

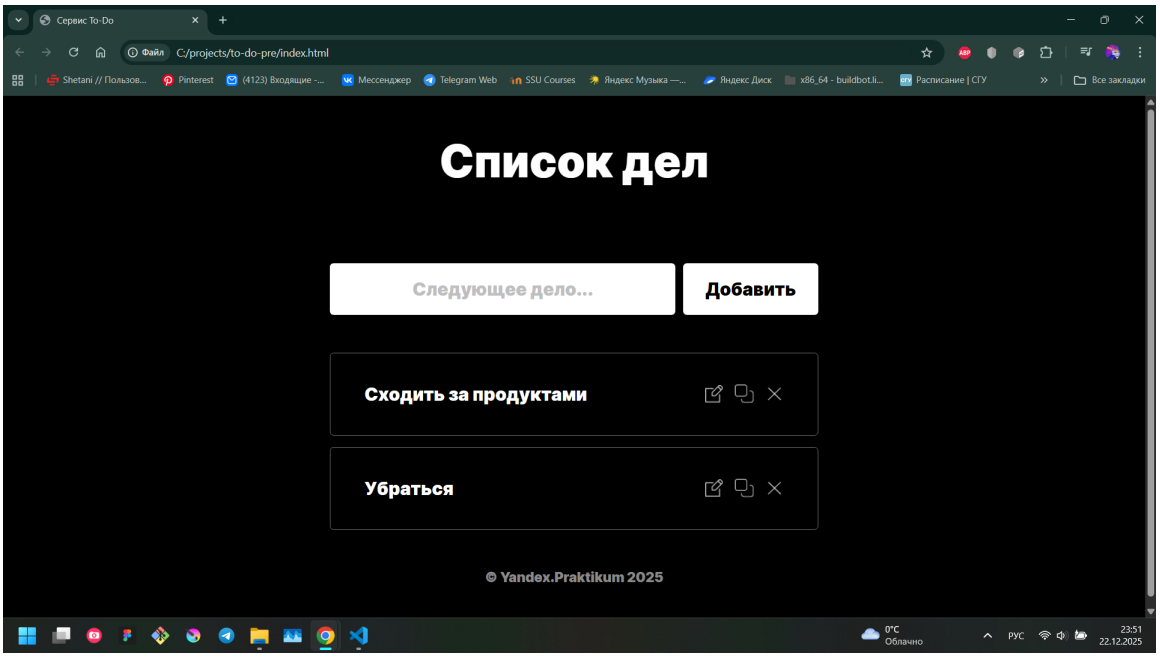

Рисунок 1 – Список дел

2. Функция createTaskElement

   Она создает и возвращает готовый DOM-элемент [4] задачи для списка дел, полностью настроенный со всеми необходимыми обработчиками событий.

```
const template = document.getElementById("to-do__item-template");
const taskElement=template.content.querySelector(".to-do__item").cloneNode(true);
```

Клонирование шаблона

- Находит HTML-шаблон с id "to-do__item-template".
- Клонирует элемент с классом .to-do__item из содержимого шаблона.
- cloneNode(true) - глубокое клонирование (со всеми дочерними элементами).

Поиск внутренних элементов

```
const taskTextElement = taskElement.querySelector(".to-do__item-text");
const deleteButton = taskElement.querySelector(".to-do__item-button_type_delete");
const duplicateButton = taskElement.querySelector(".to-do__item-button_type_duplicate");
const editButton = taskElement.querySelector(".to-do__item-button_type_edit");
```

Находит все необходимые элементы внутри клонированного шаблона:

- Элемент для текста задачи
- Кнопку удаления
- Кнопку дублирования
- Кнопку редактирования

Установка текста задачи

taskTextElement.textContent = taskText;

Устанавливает переданный текст задачи в соответствующий элемент

Назначение обработчиков событий на кнопки

```
deleteButton.addEventListener('click', () => {
  removeTask(taskElement);
});
```

При клике на кнопку удаления вызывает функцию removeTask()

Дублирование задачи:

```
duplicateButton.addEventListener('click', () => {
  duplicateTask(taskTextElement);
});
```

Редактирование задачи:

```
editButton.addEventListener('click', () => {
  enableTaskEditing(taskTextElement);
});
```

Обработчики для редактирования текста
Завершение редактирования (потеря фокуса):

```
taskTextElement.addEventListener('blur', () => {
  finishTaskEditing(taskTextElement);
});
```

Завершение редактирования (клавиша Enter):

```
taskTextElement.addEventListener('keydown', (event) => {
  if (event.key === 'Enter') {
    event.preventDefault();
    taskTextElement.blur();
  }
});
```

Возврат готового элемента
return taskElement;

2 Создание и сохранение задач

Установлен слушатель на событие [5] submit [6] формы и написан обработчик. Элемент формы уже доступен через переменную formElement.

Внутри обработчика:

- Отключена перезагрузка страницы при отправке формы.
- Получение текста задачи из поля ввода. Элемент поля ввода доступен через переменную inputElement.
- Создание готовой разметки элемента задачи с помощью функции addNewTask() и добавление её в начало контейнера .to-do__list с помощью метода prepend. Контейнер доступен через переменную listElement.
- Очищение поля ввода.

Функция getAllTasks - собирает список задач из текущей разметки и возвращает его в виде массива строк.

```
function getAllTasks() {
  const taskElements = listElement.querySelectorAll('.to-do__item-text');
  const currentTasks = [];

  taskElements.forEach(element => {
    currentTasks.push(element.textContent);
  });

  return currentTasks;
}
```

Функция updateStoredTasks - сохраняет в локальное хранилище [7] переданный в параметре массив строк задач

```
function updateStoredTasks() {
  const currentTasks = getAllTasks();
  localStorage.setItem('items', JSON.stringify(currentTasks));
}
```

Пример создания задачи. После нажатие кнопки "Добавить"задача попадет в список [8]. Смотреть рисунок 2

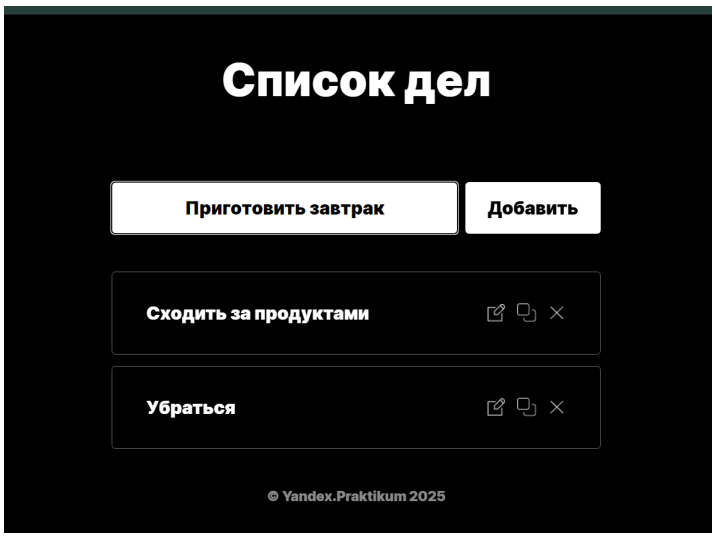

Рисунок 2 – Создание задачи

Сохранение задачи. Смотреть рисунок 3.

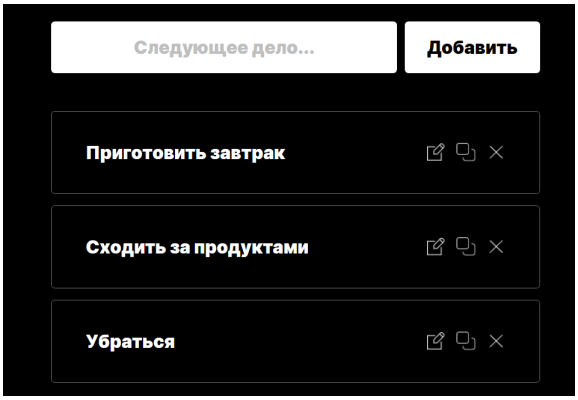

Рисунок 3 – Сохраненная задача

3 Удаление задачи

Добавлен обработчик события click. Кнопка удаления(см. рисунок 4) задачи уже доступна через переменную deleteButton.

Внутри обработчика:

— Удаление текущего элемента задачи [9], с помощью метода remove
— Создание переменной items. Результат выполнения функции getAllTasks
— Сохранение в локальное хранилище с помощью вызова функции updateStoredTasks

Рисунок 4 – Кнопка удаления

Результат кнопки "Удалить-5.

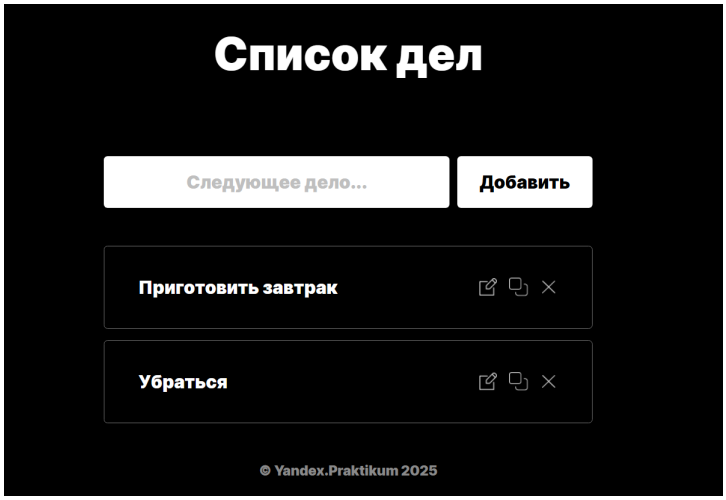

Рисунок 5 – Результат удаления

4 Копирование задачи

К кнопке с классом .to-do__item-button_type_duplicate [10] добавлен обработчик событий click(см. рисунок 6).

Рисунок 6 – Кнопка "Копировать"

Внутри обработчика:

- Получение текста текущей задачи из элемента taskTextElement
- Создание нового элемента задачи с помощью функции createTaskElement, передавая текущий текст
- Добавление новой задачи в начало списка с помощью метода prepend
- Обновление данных в локальном хранилище с помощью функции updateStoredTasks

Пример кода функции duplicateTask:

```
function duplicateTask(taskTextElement) {
  const currentText = taskTextElement.textContent;
  const newTaskElement = createTaskElement(currentText);
  listElement.prepend(newTaskElement);
  updateStoredTasks();
}
```

Результат копирования представлен на рисунке 7

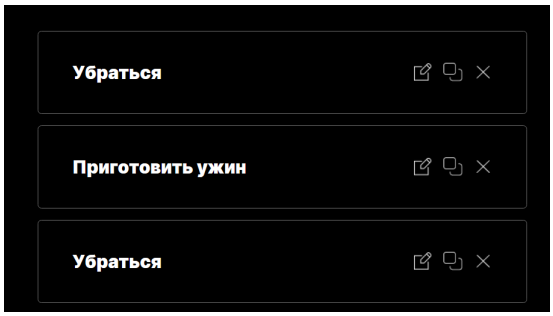

Рисунок 7 – Результат копирования

5 Редактирование задачи

Для редактирования задачи реализованы две функции:

— Функция enableTaskEditing - включает режим редактирования для элемента задачи

— Функция finishTaskEditing - завершает редактирование и сохраняет изменения

Кнопка для редактирования(см. рисунок 8)

Рисунок 8 – Кнопка "Редактировать"

Пример кода функций:

```
function enableTaskEditing(taskTextElement) {
  taskTextElement.setAttribute('contenteditable', 'true');
  taskTextElement.focus();
}

function finishTaskEditing(taskTextElement) {
  taskTextElement.setAttribute('contenteditable', 'false');
  updateStoredTasks();
}
```

Пример редактирования задачи представлен на рисунках 9 и 10

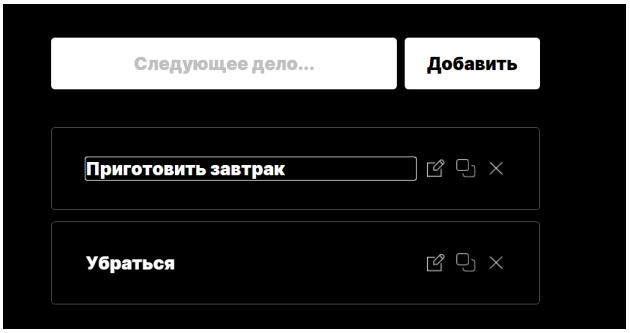

Рисунок 9 – Редактирование задачи

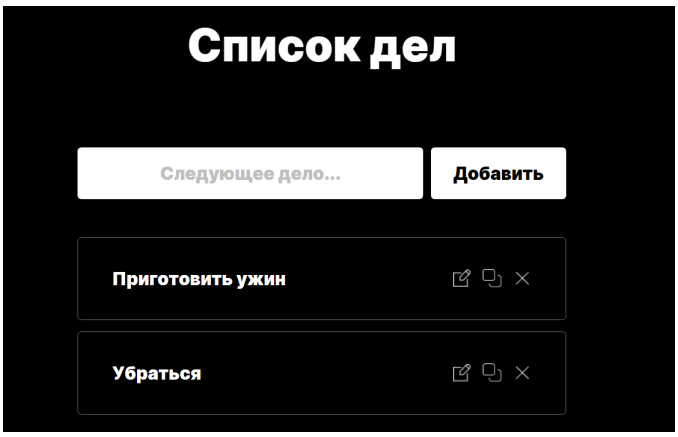

Рисунок 10 – Отредактированная задача

ЗАКЛЮЧЕНИЕ

В ходе выполнения проектной работы было разработано одностраничное веб-приложение "Сервис To-do"с полным набором функций для управления задачами. Приложение позволяет:

- Просматривать список текущих задач
- Добавлять новые задачи через форму ввода
- Удалять существующие задачи
- Дублировать задачи для быстрого создания похожих записей
- Редактировать текст задач непосредственно в интерфейсе
- Автоматически сохранять все изменения в локальное хранилище браузера

Были успешно реализованы все основные функции управления задачами, а также обеспечено сохранение данных между сессиями. Приложение имеет интуитивно понятный интерфейс и отвечает современным требованиям к веб-приложениям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Проектная работа. Сервис To-do. — URL: clck.ru/3QxmBL (дата обращения: 21.12.2025). [Электронный ресурс].

2 Хавербеке Марейн. Выразительный JavaScript. Современное веб-программирование. 3-е издание. — Санкт-Петербург : Питер, 2023.

3 Массивы. — URL: https://learn.javascript.ru/array (дата обращения: 23.12.2025). [Электронный ресурс].

4 Очень похожи на массивы, но не массивы. Коллекции в DOM. — URL: clck.ru/3QxuQW (дата обращения: 21.12.2025). [Электронный ресурс].

5 Как работают события. — URL: clck.ru/3Qxuiz (дата обращения: 21.12.2025). [Электронный ресурс].

6 Отправка формы. Событие submit. — URL: clck.ru/3Qxv3C (дата обращения: 22.12.2025). [Электронный ресурс].

7 Что такое хранилище в браузере. Какие бывают виды и чем отличаются. — URL: clck.ru/3QxudK (дата обращения: 20.12.2025). [Электронный ресурс].

8 Добавление элементов на страницу. — URL: clck.ru/3Qxv7o (дата обращения: 21.12.2025). [Электронный ресурс].

9 Удаление и перемещение элементов. — URL: clck.ru/3QxuaK (дата обращения: 22.12.2025). [Электронный ресурс].

10 Клонирование элементов. — URL: clck.ru/3Qxupq (дата обращения: 21.12.2025). [Электронный ресурс].

ПРИЛОЖЕНИЕ А

Flash-носитель с отчетом о выполненной работе

На приложенном flash-накопителе можно ознакомиться со следующими файлами:

Папка Fronted Report — LaTeX- вариант проектной практики;

Папка to-do-pre — весь исходный код;

main.pdf — отчет о выполненной проектной практики.