**Государственное бюджетное профессиональное образовательное учреждение Московской области «Колледж «Подмосковье»**

**Курсовая работа**

**Тема: создание сайта «Трекер привычек» на HTML**

Выполнила   
студентка 2 курса Геранская Ульяна Станиславовна,

по специальности / направлению Информационные технологии и программирование

Принял(а)  
Кустов Алексей Ильич

Курсовая работа принята «\_\_» \_\_\_\_2025 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Решетниково, 2025

# Содержание

[Содержание 1](#_Toc212544761)

[Введение 2](#_Toc212544762)

[1. Основы HTML 4](#_Toc212544763)

[1.1. Роль HTML в веб-разработке: история и эволюция. 4](#_Toc212544764)

[1.2. Семантическая разметка HTML: header, nav, main, article, section, aside, footer. Преимущества использования. 4](#_Toc212544765)

[1.3. Основные элементы HTML: заголовки (h1-h6), параграфы (p), списки (ul, ol, li), ссылки (a), изображения (img), таблицы (table). 5](#_Toc212544766)

[1.4. Формы в HTML: form, input (типы: text, checkbox, radio, date, submit), label, button, textarea. Важность для интерактивности. 6](#_Toc212544767)

[1.5. Атрибуты HTML-элементов: id, class, src, href, alt, title и их применение. 7](#_Toc212544768)

[2. Концепция "Трекера привычек" 8](#_Toc212544769)

[2.1. Что такое трекер привычек и его преимущества. 8](#_Toc212544770)

[2.2. Основные функции и компоненты трекера привычек (добавление привычек, отслеживание выполнения, статистика). 8](#_Toc212544771)

[2.3. Анализ существующих решений: обзор популярных трекеров привычек (преимущества и недостатки). 9](#_Toc212544772)

[Глава 3. Проектирование и разработка структуры сайта "Трекер привычек" 10](#_Toc212544773)

[3.1. Архитектура сайта 10](#_Toc212544774)

[3.2. Общая структура страниц: главная страница (дашборд), страница добавления привычки, страница статистики (если планируется). 10](#_Toc212544775)

[3.3. Взаимосвязь между страницами. 10](#_Toc212544776)

[4. Разработка HTML-структуры для главной страницы (Дашборд) 11](#_Toc212544777)

[4.1. Использование семантических тегов для организации контента (header, main, footer). 11](#_Toc212544778)

[4.2. Структурирование списка привычек: применение списков (ul, li) или таблиц (table) для отображения привычек и их статуса. 11](#_Toc212544779)

[4.3. Реализация элементов для отслеживания выполнения привычки (например, чекбоксы или кнопки). 12](#_Toc212544780)

[4.4. Отображение прогресса и статистики (если применимо на главной). 13](#_Toc212544781)

[4.5. Разработка навигационного меню (nav) для перехода по сайту. 13](#_Toc212544782)

[5. Разработка HTML-структуры для страницы добавления привычки 14](#_Toc212544783)

[5.1. Создание формы (form) для ввода данных о новой привычке. 14](#_Toc212544784)

[5.2. Использование различных типов полей ввода (input) для названия привычки, частоты, описания и т.д. 14](#_Toc212544785)

[5.3. Применение label для доступности и удобства заполнения формы. 15](#_Toc212544786)

[5.4. Создание кнопки для сохранения новой привычки (button или input type="submit"). 15](#_Toc212544787)

[Глава 6. Тестирование и демонстрация работы сайта 16](#_Toc212544788)

[6.1. Тестирование HTML-структуры 16](#_Toc212544789)

[6.2. Проверка отображения на различных браузерах (кросс-браузерность). 16](#_Toc212544790)

[6.3. Демонстрация функционала 16](#_Toc212544791)

[6.4. Пошаговая демонстрация работы сайта: добавление привычки, отметка выполнения, просмотр (если реализовано) статистики. 16](#_Toc212544792)

[6.5. Описание реального пользовательского сценария. 17](#_Toc212544793)

[7. Стилизация и адаптивный дизайн сайта "Трекер привычек" (CSS) 18](#_Toc212544794)

[7.1. Основы CSS3 и его роль в веб-разработке 18](#_Toc212544795)

[7.2. Разработка цветовой схемы и типографики 18](#_Toc212544796)

[7.3. Создание раскладки элементов с помощью Flexbox и Grid 19](#_Toc212544797)

[7.4. Реализация адаптивного дизайна 19](#_Toc212544798)

[7.5. Базовая стилизация форм и интерактивных элементов 19](#_Toc212544799)

[8. Интерактивность и динамический функционал сайта (JavaScript) 20](#_Toc212544800)

[8.1. Роль JavaScript в веб-приложениях 20](#_Toc212544801)

[8.2. Сохранение данных пользователя с помощью localStorage 20](#_Toc212544802)

[8.3. Динамическое добавление, удаление и отметка привычек 20](#_Toc212544803)

[8.4. Валидация форм на стороне клиента 21](#_Toc212544804)

[8.5. Обновление статистики в реальном времени 21](#_Toc212544805)

[9. Серверная часть (Backend) и перспективы масштабирования 22](#_Toc212544806)

[9.1. Необходимость серверной разработки 22](#_Toc212544807)

[8.2. Основные функции Backend-а 22](#_Toc212544808)

[8.3. Технологии для Backend-разработки 22](#_Toc212544809)

[8.4. Пример Backend-архитектуры 23](#_Toc212544810)

[10. Расширенный функционал и дальнейшее развитие 24](#_Toc212544811)

[10.1. Расширение возможностей трекера привычек 24](#_Toc212544812)

[10.2. Тестирование и оптимизация 24](#_Toc212544813)

[Заключение 26](#_Toc212544814)

[Приложения 28](#_Toc212544815)

[Список использованной литературы 29](#_Toc212544816)

# ****Введение****

В мире программирования существует постоянная потребность в разработке инструментов, способных модифицировать и расширять функциональность приложений. Одним из мощных методов достижения этой цели является создание низкоуровневых библиотек, предоставляющих программистам инструменты для внедрения изменений в работу приложений на уровне исполняемого кода.

Данный проект нацелен на разработку такой низкоуровневой библиотеки на языке программирования C++ для операционной системы Windows. Библиотека предоставит разработчикам инструменты для модификации и расширения функциональности приложений, а также создания хуков - механизмов перехвата вызовов функций.

Мы предлагаем исследовать область создания модульных расширений, позволяющих внедрять новый функционал в существующие приложения без необходимости изменения исходного кода. Этот подход дает возможность создавать персонализированные и улучшенные версии программ без необходимости изменения исходного кода исследуемого приложения.

В данной курсовой работе мы будем рассматривать основные принципы разработки низкоуровневых библиотек, механизмы работы с памятью и функциями операционной системы, необходимые для создания библиотеки модификации приложений. Также будут рассмотрены техники создания хуков и их использование в контексте модинга.

Основная цель курсовой - предоставить читателю полное представление о создании и использовании библиотек для модификации приложений на C++ и продемонстрировать применение этих знаний на практике.

# Основы HTML

## 1.1. Роль HTML в веб-разработке: история и эволюция.

HyperText Markup Language (HTML) является краеугольным камнем Всемирной паутины. Его история берет начало в начале 1990-х годов, когда Тим Бернерс-Ли разработал первую версию языка для обмена научными документами. С тех пор HTML претерпел значительные изменения, эволюционируя от относительно простого языка разметки документов к мощному инструменту для создания сложных, динамичных и интерактивных веб-приложений. Каждая новая версия HTML привносила улучшения, направленные на повышение семантической точности, улучшение доступности, поддержку мультимедиа и упрощение разработки. Последние стандарты, в частности HTML5, существенно расширили возможности фронтенд-разработчиков, предоставив новые элементы для семантической разметки, встроенные API для работы с мультимедиа, графикой и данными, а также улучшенную поддержку форм и валидации. HTML5 стал де-факто стандартом, обеспечивающим совместимость и единообразие веб-контента на различных устройствах и платформах.

## 1.2. Семантическая разметка HTML: header, nav, main, article, section, aside, footer. Преимущества использования.

HTML ввел набор семантических элементов, которые позволяют разработчикам более точно описывать назначение и структуру контента страницы. В отличие от универсальных контейнеров вроде **<div>** и **<span>**, семантические теги несут смысловую нагрузку, улучшая понимание структуры документа как браузерами, так и поисковыми системами, а также вспомогательными технологиями (например, программами чтения с экрана для людей с нарушениями зрения). **header**: Определяет вводную часть документа или секции (например, заголовок сайта, логотип, навигация). **nav**: Представляет собой блок навигации, содержащий ссылки на другие страницы или разделы сайта. **main**: Обозначает основной, уникальный контент страницы. На одной странице должен быть только один элемент **main**. **article**: Обозначает самостоятельный, независимый блок контента, который может быть опубликован отдельно (например, пост в блоге, новостная статья). **section**: Определяет логически обособленную часть документа, которая обычно имеет заголовок. **aside**: Представляет контент, который косвенно связан с основным контентом страницы (например, боковая панель, рекламный блок). **footer**: Определяет нижний колонтитул документа или секции (например, информация об авторских правах, контактные данные).

Использование семантической разметки дает значительные преимущества: улучшение SEO-оптимизации за счет лучшего понимания контента поисковыми системами, повышение доступности веб-ресурсов для пользователей с ограниченными возможностями, упрощение поддержки и модификации кода, а также более предсказуемое поведение кода в различных браузерах.

## 1.3. Основные элементы HTML: заголовки (h1-h6), параграфы (p), списки (ul, ol, li), ссылки (a), изображения (img), таблицы (table).

Для построения содержимого веб-страниц HTML5 предоставляет набор базовых элементов: **Заголовки (h1 - h6)**: Используются для структурирования контента по иерархии. **<h1>**является самым важным заголовком (обычно для названия страницы), а **<h6>** – наименее важным. Правильное использование заголовков критично для SEO и доступности. **Параграфы (p)**: Предназначены для обозначения абзацев текста. **Списки**: **<ul>** (Unordered List): Используется для неупорядоченных списков (маркированных). **<ol>** (Ordered List): Используется для упорядоченных списков (нумерованных или с использованием букв). **<li>** (List Item): Обозначает отдельный элемент списка. **Ссылки (a)**: Элемент "якорь", позволяющий создавать гиперссылки на другие веб-страницы, файлы или разделы текущей страницы. Атрибут **href** указывает на целевой ресурс. **Изображения (img)**: Встраивает изображения в документ. Атрибут **src** указывает путь к файлу изображения, а **alt** – альтернативный текст, который отображается, если изображение не загрузилось, и используется поисковыми системами. **Таблицы (table)**: Используются для представления табличных данных. Включают в себя элементы **<thead>** (заголовок таблицы), **<tbody>** (тело таблицы), **<tr>** (строка таблицы), **<th>** (заголовок столбца/строки) и **<td>** (ячейка данных).

## 1.4. Формы в HTML: form, input (типы: text, checkbox, radio, date, submit), label, button, textarea. Важность для интерактивности.

Формы являются основным механизмом взаимодействия пользователя с веб-сайтом, позволяя отправлять данные на сервер. **<form>**: Обертка для всех элементов формы. Атрибуты **action** (URL для отправки данных) и **method** (HTTP-метод, обычно **GET** или **POST**) определяют, куда и как будут отправлены данные. **<input>**: Универсальный элемент для ввода данных. HTML5 значительно расширил его возможности, добавив множество типов: \* **text**: Однострочное текстовое поле. **checkbox**: Флажок (для выбора одного или нескольких вариантов). **radio**: Маркированный переключатель (для выбора одного варианта из группы). **date**: Поле для выбора даты. **submit**: Кнопка для отправки формы. И многие другие (**email, password, number, url**и т.д.). **<label>**: Связывает текстовую метку с элементом формы (например, с **input**). Атрибут **for** у **<label>**должен совпадать с **id** соответствующего элемента формы, что улучшает доступность и удобство использования (клик по метке активирует связанный элемент). **<button>**: Более гибкая альтернатива **input type="submit"**, позволяющая задавать различные типы действий (**submit, reset, button**). **<textarea>**: Многострочное текстовое поле для ввода большого объема текста.

Формы критически важны для создания интерактивных веб-приложений, поскольку именно они позволяют пользователю вносить информацию, настраивать параметры и отправлять запросы.

## 1.5. Атрибуты HTML-элементов: id, class, src, href, alt, title и их применение.

Атрибуты предоставляют дополнительную информацию об HTML-элементах и управляют их поведением или отображением. **id**: Уникальный идентификатор элемента на странице. Может использоваться для нацеливания на элемент с помощью CSS или JavaScript, а также для создания ссылок-якорей. Каждый **id** должен быть уникальным в пределах документа. **class**: Идентификатор, который может быть присвоен нескольким элементам. Позволяет группировать элементы для применения общих стилей (CSS) или выполнения одних и тех же действий (JavaScript). **src**(source): Используется в элементах, которые ссылаются на внешний ресурс, например, **<img>**(путь к изображению) или **<script>**(путь к файлу скрипта). **href**(hypertext reference): Используется в теге **<a>** для указания URL-адреса, на который ведет ссылка, или в **<link>**для указания пути к файлу стилей. **alt**(alternative text): Обязательный атрибут для тега **<img>**, предоставляющий текстовое описание изображения. Важен для доступности и SEO. **title**: Предоставляет дополнительную информацию об элементе, которая обычно отображается при наведении курсора мыши.

# 2. Концепция "Трекера привычек"

## 2.1. Что такое трекер привычек и его преимущества.

Трекер привычек – это инструмент, который помогает пользователям систематически отслеживать выполнение запланированных действий, направленных на формирование новых или поддержание существующих полезных привычек. Его основная ценность заключается в визуализации прогресса, что является мощным мотиватором. Преимущества использования трекера привычек включают: **Повышение осознанности:** Пользователь лучше понимает, насколько регулярно он следует своим целям. **Визуализация прогресса:** Наглядное отображение выполненных и пропущенных дней помогает увидеть динамику и оценить достигнутое. **Мотивация:** Видение "цепочки" успешно выполненных дней стимулирует не прерывать ее. **Ответственность:** Сам факт отслеживания повышает ответственность за свои действия. **Выявление закономерностей:** Позволяет понять, в какие дни или при каких условиях привычка выполняется легче или сложнее.

## 2.2. Основные функции и компоненты трекера привычек (добавление привычек, отслеживание выполнения, статистика).

Типичный трекер привычек включает следующие ключевые компоненты и функции: **Добавление привычки:** Пользователь должен иметь возможность ввести название новой привычки, возможно, указать ее частоту (ежедневно, несколько раз в неделю), установить цель (например, "выполнить 30 раз"). **Отслеживание выполнения:** Основная функция, где пользователь отмечает, выполнил ли он привычку в текущий день. Это может быть реализовано в виде чекбоксов, кнопок или свайпов. **Отображение списка привычек:** Наглядное представление всех отслеживаемых привычек, часто с индикаторами прогресса или текущего статуса. **Статистика и отчетность:** Визуализация данных о выполнении привычек за различные периоды (день, неделя, месяц, год) в виде графиков, календарей выполнения, процентного соотношения успехов. **Напоминания:** Уведомления, помогающие не забыть о выполнении привычки. **Кастомизация:** Возможность настройки внешнего вида, цветовых схем, выбора иконок для привычек.

## 2.3. Анализ существующих решений: обзор популярных трекеров привычек (преимущества и недостатки).

Рынок приложений для отслеживания привычек изобилует различными решениями, как для мобильных платформ, так и в виде веб-сервисов. Среди популярных примеров можно назвать:

**Streaks:** Минималистичное приложение для iOS, фокусирующееся на поддержании непрерывных серий выполнения привычек. Преимущества: простота, элегантный дизайн. Недостатки: ограниченность платформой, отсутствие веб-версии.

**Loop Habit Tracker (Android):** Бесплатное приложение с открытым исходным кодом, предлагающее богатый функционал и статистику. Преимущества: высокая степень кастомизации, подробная аналитика, отсутствие рекламы. Недостатки: только для Android.

**TickTick, Todoist (частично):** Хотя это в первую очередь менеджеры задач, они часто включают функции отслеживания привычек. Преимущества: интеграция с задачами. Недостатки: отслеживание привычек может быть не таким глубоким, как в специализированных приложениях.

Анализ этих решений показывает, что пользователи ценят как простоту и минимализм, так и продвинутую статистику и геймификацию. Для данной курсовой работы мы ориентируемся на создание базовой, но семантически корректной структуры, которая может быть расширена любым из этих подходов.

# Глава 3. Проектирование и разработка структуры сайта "Трекер привычек"

## 3.1. Архитектура сайта

## 3.2. Общая структура страниц: главная страница (дашборд), страница добавления привычки, страница статистики (если планируется).

Для обеспечения полноты функционала "Трекера привычек" планируется следующая базовая архитектура сайта: **Главная страница (Дашборд):** Центральный элемент, который отображает список всех отслеживаемых пользователем привычек. На этой странице пользователь сможет видеть текущий статус выполнения каждой привычки за день, а также иметь быстрый доступ к инструментам отметки выполнения. Здесь же может быть представлена краткая сводка прогресса. **Страница добавления привычки:** Отдельный раздел, предназначенный для ввода данных о новой привычке. Пользователь сможет указать название, описание, частоту и другие необходимые параметры. **Страница статистики (опционально, но желательно для полноценного трекера):** Этот раздел будет посвящен детальному анализу прогресса. Здесь может отображаться календарь выполнения, графики успешности по каждой привычке, общая статистика.

## 3.3. Взаимосвязь между страницами.

Навигация между этими разделами будет осуществляться посредством ссылок, расположенных в главном меню или в соответствующих блоках интерфейса. Например, с главной страницы пользователь сможет перейти на страницу добавления новой привычки, а со страницы добавления – вернуться на дашборд. Страница статистики будет доступна как с главной, так и, возможно, с индивидуальных страниц просмотра привычек. Целостность пользовательского опыта будет обеспечена последовательным дизайном и единообразным размещением навигационных элементов.

# Разработка HTML-структуры для главной страницы (Дашборд)

## 4.1. Использование семантических тегов для организации контента (header, main, footer).

Структура главной страницы, или дашборда, будет построена с применением семантических тегов HTML5 для обеспечения максимальной читаемости кода и лучшей индексации. **<header>**: Будет содержать заголовок сайта ("Трекер привычек") и, возможно, элемент навигации (**<nav>**). **<nav>**: Внутри **<header>**или как отдельный элемент, будет содержать ссылки на другие разделы сайта (например, "Добавить привычку", "Статистика"). **<main>**: Является основным контейнером для всей информации, относящейся к дашборду. Внутри него будет располагаться список привычек. **<footer>**: Будет содержать информацию об авторских правах, контактные данные или ссылки на дополнительные ресурсы.

## 4.2. Структурирование списка привычек: применение списков (ul, li) или таблиц (table) для отображения привычек и их статуса.

Для отображения списка привычек предлагается использовать семантически корректные элементы. \* Вариант 1 (список): Использование неупорядоченного списка **<ul>**, где каждый элемент **<li>** будет представлять отдельную привычку. Внутри **<li>** будут располагаться элементы для отслеживания статуса (например, чекбокс), названия привычки и, возможно, краткой статистики. Этот вариант предпочтителен, если привычки имеют простую линейную структуру:

**html <main> <h2>**Мои привычки**</h2>** **<ul** **id="habit-list"> <li> <input type="checkbox" id="habit1-done"> <label for="habit1-done">**Утренняя зарядка**</label> <!--** Дополнительные элементы, например, кнопка настройки **--> </li> <li> <input type="checkbox" id="habit2-done"> <label for="habit2-done">Чтение 30 минут</label> </li> <!-- ... другие привычки --> </ul> </main>**

Вариант 2 (таблица): Для более сложной структуры, где важна четкая колоночная организация (например, название привычки, частота, индикатор выполнения за каждый день недели), может быть использована таблица:

**<table>. html <main> <h2>**Мои привычки**</h2> <table id="habit-table"> <thead> <tr> <th>**Привычка**</th> <th>**Понедельник**</th> <th>**Вторник**</th> <th>...</th> </tr> </thead> <tbody> <tr> <td>**Утренняя зарядка**</td> <td><input type="checkbox"></td> <td><input type="checkbox"></td> <td>...</td> </tr> <!-- ...** другие привычки **--> </tbody> </table> </main>**

Выбор между списком и таблицей зависит от сложности данных и требуемого представления. Для нашего базового варианта, где основной фокус на отметке выполнения, список **<ul>** с вложенными **<li>** будет более гибким и семантически оправданным.

## 4.3. Реализация элементов для отслеживания выполнения привычки (например, чекбоксы или кнопки).

Для отметки выполнения привычки наиболее интуитивным и распространенным элементом является чекбокс **(<input type="checkbox">).** Каждый чекбокс будет ассоциирован с конкретной привычкой и конкретным днем (или просто означать общее выполнение).

<li>

<input type="checkbox" id="habit- зарядка-done" class="habit-checkbox">

<label for="habit-зарядка-done">Утренняя зарядка</label>

<span class="habit-streak">Серия: 5 дней</span>

<!-- Кнопки управления -->

<button class="edit-habit-btn">Изменить</button>

<button class="delete-habit-btn">Отмена</button>

</li>

Каждый чекбокс будет иметь уникальный **id** (для связи с **<label>**) и **class** (для групповой обработки JavaScript). Наличие **label** обеспечивает доступность. Дополнительно могут быть добавлены кнопки для редактирования или удаления привычки.

## 4.4. Отображение прогресса и статистики (если применимо на главной).

На главной странице можно предусмотреть место для краткой сводки прогресса, например, общего количества выполненных привычек за день или текущую неделю. Это может быть простой параграф **<p>** или **<span>** с соответствующим классом.

**<div class="daily-summary">**

**<p>**Выполнено привычек сегодня: **<span id="completed-today">0</span>** из **<span id="total-habits">3</span></p>**

**</div>**

Более детальная статистика будет вынесена на отдельную страницу, но основы для ее отображения (например, структура для будущих графиков) могут быть заложены уже на главной.

## 4.5. Разработка навигационного меню (nav) для перехода по сайту.

Навигационное меню, реализованное с помощью тега **<nav>,** обеспечит пользователю удобный доступ ко всем разделам сайта.

<header>

<h1>Трекер привычек</h1>

<nav>

<ul>

<li><a href="index.html">Главная</a></li>

<li><a href="add-habit.html">Добавить привычку</a></li>

<li><a href="stats.html">Статистика</a></li>

</ul>

</nav>

</header>

Использование списка **<ul>** с ссылками **<a>** внутри **<nav>** является стандартной практикой для создания меню.

# Разработка HTML-структуры для страницы добавления привычки

## 5.1. Создание формы (form) для ввода данных о новой привычке.

Страница добавления привычки будет построена вокруг элемента **<form>**, который будет содержать все необходимые поля ввода.

<main>

<h2>Добавить новую привычку</h2>

<form id="add-habit-form">

<!-- Поля формы -->

<button type="submit">Сохранить привычку</button>

</form>

</main>

Атрибут **id="add-habit-form"** позволит легко обращаться к форме с помощью JavaScript для обработки отправки данных.

## 5.2. Использование различных типов полей ввода (input) для названия привычки, частоты, описания и т.д.

Для сбора информации о новой привычке будут использоваться разнообразные типы полей ввода **<input>** и другие элементы форм:

Название привычки: **<input type="text" id="habit-name" name="habit-name" required**> – обязательное текстовое поле.

Описание: **<textarea id="habit-description" name="habit-description"></textarea>** – многострочное поле для более подробного описания.

Частота выполнения:

Для ежедневных привычек: **<input type="checkbox" id="habit-daily" name="habit-frequency" value="daily"> <label for="habit-daily">Ежедневно</label>**

Для выбора дней недели: Можно использовать набор чекбоксов для каждого дня **(<input type="checkbox" name="day" value="mon"> Пн, <input type="checkbox" name="day" value="tue"> Вт и т.д.)** или выпадающий список **(<select>).**

Цель (например, количество повторений): **<input type="number" id="habit-goal" name="habit-goal" min="1"> \* Цвет/иконка: <input type="color" id="habit-color" name="habit-color" value="#ffffff">**

## 5.3. Применение label для доступности и удобства заполнения формы.

К каждому элементу формы будет применена соответствующая метка **<label>,** связанная с элементом формы через атрибут **for**. Это улучшает доступность (пользователи программ чтения с экрана смогут ассоциировать метку с полем) и удобство использования (клик по метке активирует соответствующее поле ввода).

<div>

<label for="habit-name">Название привычки:</label>

<input type="text" id="habit-name" name="habit-name" required>

</div>

## 5.4. Создание кнопки для сохранения новой привычки (button или input type="submit").

Завершает форму кнопка, которая инициирует процесс отправки данных.

**<button type="submit" id="save-habit-btn">Добавить привычку</button>**

Использование **type="submit"** гарантирует, что кнопка отправит форму.

# Глава 6. Тестирование и демонстрация работы сайта

## 6.1. Тестирование HTML-структуры

## 6.2. Проверка отображения на различных браузерах (кросс-браузерность).

После разработки HTML-структуры, критически важно провести тестирование на предмет кросс-браузерности. Это означает проверку того, как страница отображается в различных популярных веб-браузерах (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) и их версиях. Цель – убедиться, что все элементы располагаются корректно, формы работают как ожидалось, и нет никаких визуальных артефактов или поломок макета, которые могли бы возникнуть из-за различий в интерпретации HTML-кода браузерами. На данном этапе, когда мы работаем только с HTML, основные проверки будут касаться корректного отображения структуры, доступности ссылок и полей форм.

## 6.3. Демонстрация функционала

## 6.4. Пошаговая демонстрация работы сайта: добавление привычки, отметка выполнения, просмотр (если реализовано) статистики.

На данном этапе демонстрация будет сосредоточена на том, как HTML-структура позволяет пользователю выполнять базовые действия: Навигация: Пользователь демонстрирует переход между страницами ("Главная", "Добавить привычку", "Статистика") по ссылкам в меню **<nav>.** Добавление привычки: Пользователь показывает, как заполняется форма на странице добавления привычки: ввод названия, выбор частоты, нажатие кнопки "Добавить". Отметка выполнения: На главной странице пользователь демонстрирует, как можно отметить выполнение привычки, кликнув по чекбоксу **<input type="checkbox">.** Визуальное изменение состояния чекбокса (пустой/отмеченный) является первичной реакцией. Просмотр (если заложены элементы): Если на главной или статистике есть **placeholder-**ы для статистики (например, **<span id="completed-today">0</span>**), демонстрируется, что эти места существуют.

## 6.5. Описание реального пользовательского сценария.

Представим типичный день пользователя: 1. Пользователь заходит на сайт "Трекер привычек" (главная страница). 2. Он видит список своих привычек, например, "Утренняя зарядка", "Выпить 2 литра воды", "Прочитать 10 страниц книги". 3. После выполнения утренней зарядки, он находит соответствующий чекбокс рядом с привычкой и отмечает его. Чекбокс меняет свое состояние. 4. Вечером, перед сном, он отмечает выполнение остальных привычек. 5. Если пользователь хочет добавить новую привычку, например, "Медитация 15 минут", он переходит на страницу "Добавить привычку", заполняет форму (название "Медитация", частота "Ежедневно") и нажимает "Добавить". Новая привычка появляется в списке на главной странице. 6. В конце недели он может перейти на страницу "Статистика", чтобы увидеть, насколько успешно он следовал своим привычкам, и это будет мотивировать его продолжать.

# Стилизация и адаптивный дизайн сайта "Трекер привычек" (CSS)

## 7.1. Основы CSS3 и его роль в веб-разработке

Cascading Style Sheets (CSS) – это язык стилей, который определяет внешний вид и оформление HTML-документов. Если HTML отвечает за структуру и контент, то CSS – за его визуальное представление. CSS3, последняя версия стандарта, значительно расширила возможности стилизации, добавив поддержку анимаций, переходов, градиентов, теней, гибких (Flexbox) и блочных (Grid) раскладок, а также инструменты для создания адаптивного дизайна.

## 7.2. Разработка цветовой схемы и типографики

**Цветовая схема:** Для сайта "Трекер привычек" была выбрана спокойная, но современная цветовая палитра. Основной фон (#f4f7f6) создает ощущение легкости и чистоты. Акцентные цвета, такие как темно-синий (#2c3e50) для заголовков и навигации, и яркий синий (#3498db) для интерактивных элементов (кнопки, чекбоксы), помогают выделить важные части интерфейса и придать ему современный вид. Серые и приглушенные оттенки (#7f8c8d, #95a5a6) используются для второстепенной информации, такой как счетчики серий или кнопки отмены, чтобы не отвлекать от основного контента.

**Типографика:** Основным шрифтом выбран **Arial**, как простой, читаемый и широкодоступный шрифт без засечек. Для заголовков используются более крупные размеры и жирное начертание, чтобы создать визуальную иерархию. Межстрочный интервал (line-height: 1.6) обеспечивает комфортное чтение больших объемов текста.

## 7.3. Создание раскладки элементов с помощью Flexbox и Grid

CSS Flexbox и Grid Layout являются мощными инструментами для создания современных раскладок. **Flexbox:** Применяется для одномерного выравнивания элементов вдоль одной оси. Например, в **.habit-item** Flexbox используется для размещения чекбокса, метки, счетчика серии и кнопок действий в одну строку, с возможностью выравнивания по центру и распределения пространства. **Grid Layout:** Хотя в данном базовом примере Grid не был широко использован, он идеально подходит для создания сложных двумерных раскладок, например, сетки карточек привычек на странице статистики или для построения основного макета страницы.

## 7.4. Реализация адаптивного дизайна

Адаптивность достигается за счет использования: **Мета-тега viewport:** Обеспечивает правильное масштабирование на мобильных устройствах. **Относительных единиц:** Ширина (**width: 100%), max-width, em, rem** позволяют элементам масштабироваться пропорционально. **Медиа-запросов (@media):** Позволяют применять различные стили в зависимости от размера экрана. Например, на малых экранах (**max-width: 768px**), элементы **.habit-item** перестраиваются в вертикальный столбец, кнопки становятся на всю ширину, а заголовки уменьшаются для лучшего соответствия мобильному интерфейсу.

## 7.5. Базовая стилизация форм и интерактивных элементов

**Формы:** Поля ввода (**input, textarea, select**) стилизованы с добавлением границ, скругленных углов и **box-sizing: border-box;** для корректного расчета размеров. Метки (**label**) выровнены с полями.

**Чекбоксы и кнопки:** Их внешний вид кастомизирован с использованием **transform, accent-color**и**transition** для улучшения пользовательского опыта.

# Интерактивность и динамический функционал сайта (JavaScript)

## 8.1. Роль JavaScript в веб-приложениях

JavaScript является языком программирования, который оживляет веб-страницы, делая их интерактивными и динамичными. Он позволяет обрабатывать действия пользователя, управлять контентом, взаимодействовать с сервером и выполнять сложные вычисления. В контексте "Трекера привычек", JavaScript обеспечивает основную логику работы приложения.

## 8.2. Сохранение данных пользователя с помощью localStorage

Для того чтобы привычки и их статусы сохранялись между сессиями пользователя, используется **localStorage** браузера. **loadHabits():**При загрузке страницы, этот метод извлекает сохраненные данные из **localStorage** (по ключу **habitsTracker**), парсит их из JSON-строки в JavaScript-объект. Если данных нет, возвращается пустой массив. **saveHabits():**Этот метод сериализует текущий массив **habits** в JSON-строку и сохраняет его в **localStorage**. Вызывается после любых изменений данных (добавление, удаление, отметка выполнения).

## 8.3. Динамическое добавление, удаление и отметка привычек

Добавление: Форма на add-habit.html при отправке (обработка **handleAddHabitFormSubmit**) собирает данные, создает новый объект привычки с уникальным **id**, добавляет его в массив **habits**, сохраняет и обновляет главный список.

Отметка выполнения: **Обработчик handleCheckboxClick**реагирует на изменение состояния **чекбокса**. Он определяет, какая привычка была отмечена, и добавляет или удаляет текущую дату из массива **completedDates**. Затем происходит сохранение данных и обновление отображения серии.

Удаление: Кнопка "Удалить" вызывает функцию **deleteHabit(),** которая находит привычку по **id**, удаляет ее из массива **habits**, сохраняет изменения и перерисовывает список.

## 8.4. Валидация форм на стороне клиента

JavaScript обеспечивает первичную проверку данных, введенных пользователем, перед их отправкой или сохранением. Обязательные поля: Поле названия привычки (**habit-name**) проверяется на пустоту. Числовые поля: Цель (**habit-goal**) парсится как число, и проверяется, что она является положительной (**min="1"** в HTML также помогает, но JS-валидация надежнее). Логическая целостность: Проверяется, что выбрана хотя бы одна частота выполнения.

Это улучшает пользовательский опыт, предотвращая ввод некорректных данных и снижая нагрузку на будущий сервер.

## 8.5. Обновление статистики в реальном времени

Функции **updateSummary()** **и updateStreakDisplay()** обеспечивают моментальное обновление информации о выполненных привычках и длительности серий после каждого действия пользователя, что делает интерфейс более отзывчивым.

# 9. Серверная часть (Backend) и перспективы масштабирования

## 9.1. Необходимость серверной разработки

Хотя текущая версия сайта функциональна благодаря **localStorage**, для создания полноценного, многопользовательского приложения с возможностью синхронизации данных между устройствами и надежного хранения информации, необходима серверная часть (**Backend**).

## 8.2. Основные функции Backend-а

**Хранение данных:** Сервер будет отвечать за надежное хранение всех данных пользователя (привычки, статистика, профиль) в базе данных (например, PostgreSQL, MongoDB).

**API (Application Programming Interface):** Сервер будет предоставлять API для клиентской части (frontend), через которое будут осуществляться запросы на получение, добавление, изменение и удаление данных.

**Аутентификация и авторизация:** Система регистрации и входа пользователей для персонализации и защиты данных.

**Синхронизация:** Обеспечение доступа к актуальным данным пользователя с любого устройства.

**Обработка фоновых задач:** Реализация отправки напоминаний (push-уведомлений) о выполнении привычек.

## 8.3. Технологии для Backend-разработки

Для реализации серверной части могут быть использованы различные технологии: **Языки программирования:** Python (Django, Flask), Node.js (Express), Ruby (Rails), Java (Spring) и другие. **Базы данных:** Реляционные (PostgreSQL, MySQL) или NoSQL (MongoDB, Cassandra). **Облачные платформы:** AWS, Google Cloud, Azure для хостинга и масштабирования.

## 8.4. Пример Backend-архитектуры

Типичная архитектура будет включать: **Клиент (Frontend):** HTML, CSS, JavaScript (React, Vue, Angular или чистый JS). **API Gateway:** Управляет запросами к микросервисам или монолитному backend-приложения. **Backend-сервисы:** Обрабатывают бизнес-логику. **База данных:** Хранит информацию.

# 10. Расширенный функционал и дальнейшее развитие

## 10.1. Расширение возможностей трекера привычек

Текущая версия является базовой, но существует множество путей для улучшения и добавления нового функционала:

**Установка напоминаний:** Возможность установить конкретное время для напоминания о выполнении привычки.

**Создание групп привычек:** Группировка привычек по категориям (например, "Здоровье", "Работа", "Личное развитие").

**Продвинутая статистика и графики:** Визуализация данных в виде различных графиков (линейные, столбчатые, календари выполнения), расчет более сложных метрик (например, среднее время выполнения, корреляция между привычками).

**Геймификация:** Добавление элементов игры, таких как очки, уровни, достижения, "полоски" успеха, чтобы повысить мотивацию.

**Социальные функции:** Возможность делиться прогрессом с друзьями, создавать группы поддержки, участвовать в челленджах.

**Интеграция с другими сервисами:** Синхронизация с фитнес-трекерами, календарями, приложениями для медитации.

**Виртуальный ассистент:** Интеграция с AI-моделями для анализа прогресса, выдачи рекомендаций и мотивационных сообщений.

## 10.2. Тестирование и оптимизация

**Углубленное тестирование:** Помимо базовой кросс-браузерной и адаптивной проверки, необходимо провести:

**Функциональное тестирование:** Проверка всех функций на корректность работы.

**Тестирование производительности:** Измерение скорости загрузки и отклика приложения, особенно при большом количестве привычек и данных.

**Нагрузочное тестирование (для Backend):** Проверка, как сервер справляется с одновременными запросами от множества пользователей.

**Юзабилити-тестирование:** Привлечение реальных пользователей для получения обратной связи по удобству использования.

**Оптимизация производительности:**

**Frontend:** Минимизация CSS и JavaScript файлов, оптимизация изображений, ленивая загрузка (lazy loading).

**Backend:** Оптимизация запросов к базе данных, кеширование данных.

**SEO-оптимизация:** Для публичных страниц (если таковые будут) – настройка мета-тегов, семантической разметки, структуры URL.

# Заключение

В рамках данной курсовой работы был выполнен полный цикл проектирования и разработки базовой версии веб-сайта "Трекер привычек". Работа началась с теоретического анализа основ HTML и концепции трекеров привычек, что позволило определить ключевые требования к будущему приложению.

На этапе проектирования была разработана архитектура сайта, включающая главную страницу (дашборд), страницу добавления привычки и задел под страницу статистики. Основное внимание было уделено созданию семантически корректной HTML-структуры, что обеспечивает лучшую доступность, индексацию поисковыми системами и простоту дальнейшей разработки.

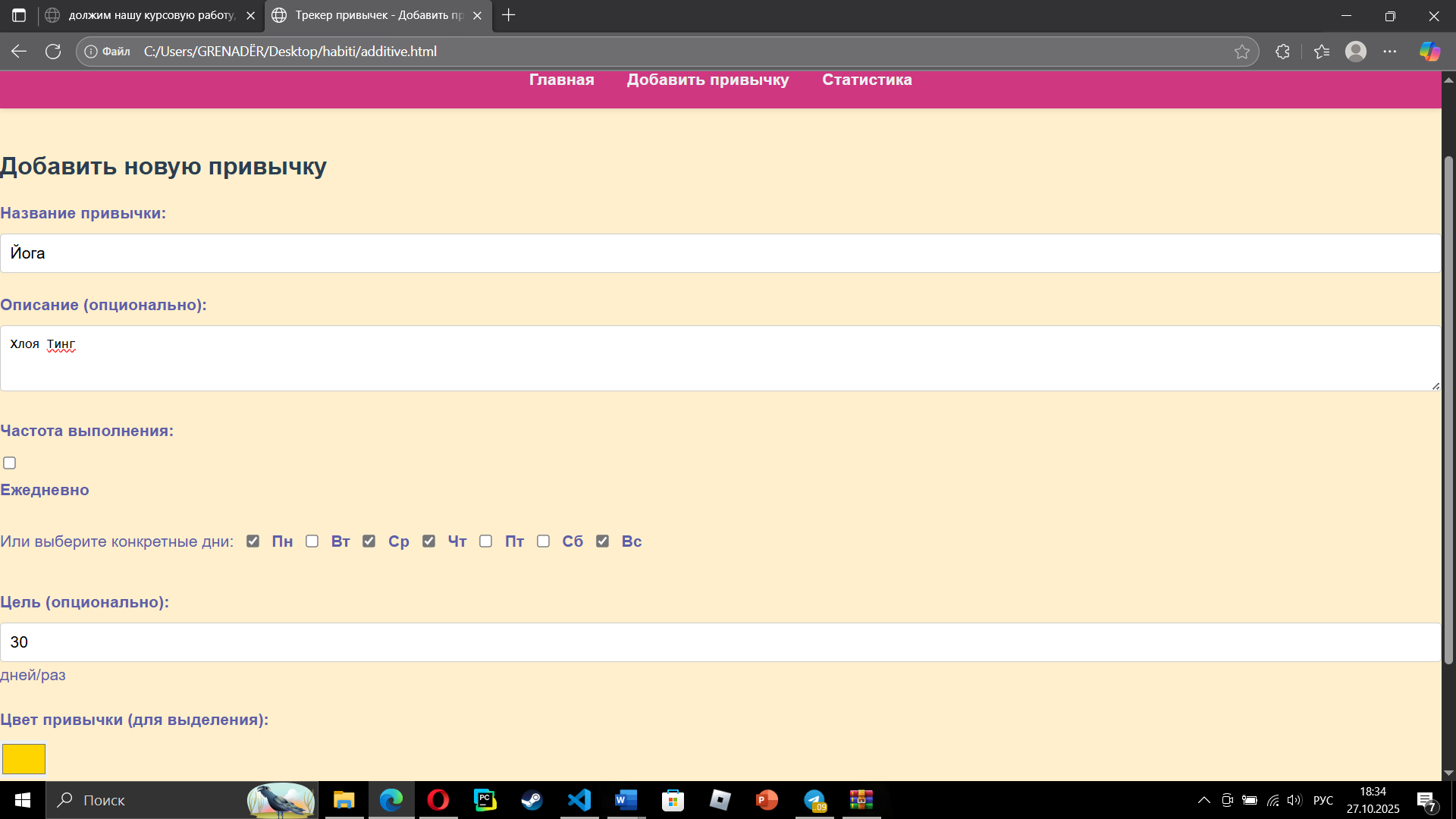
Последующая глава была посвящена стилизации сайта с использованием CSS3. Была разработана привлекательная цветовая схема и типографика, реализована адаптивная раскладка элементов с применением Flexbox, что гарантирует корректное отображение на устройствах с любыми размерами экранов.

Ключевой этап работы заключался в реализации интерактивности с помощью JavaScript. Были успешно внедрены функции добавления, удаления и отметки выполнения привычек, сохранение данных пользователя в localStorage для обеспечения их персистентности, а также базовая валидация форм на стороне клиента. Разработаны алгоритмы для расчета текущей серии выполнения привычки и отображения сводной статистики.

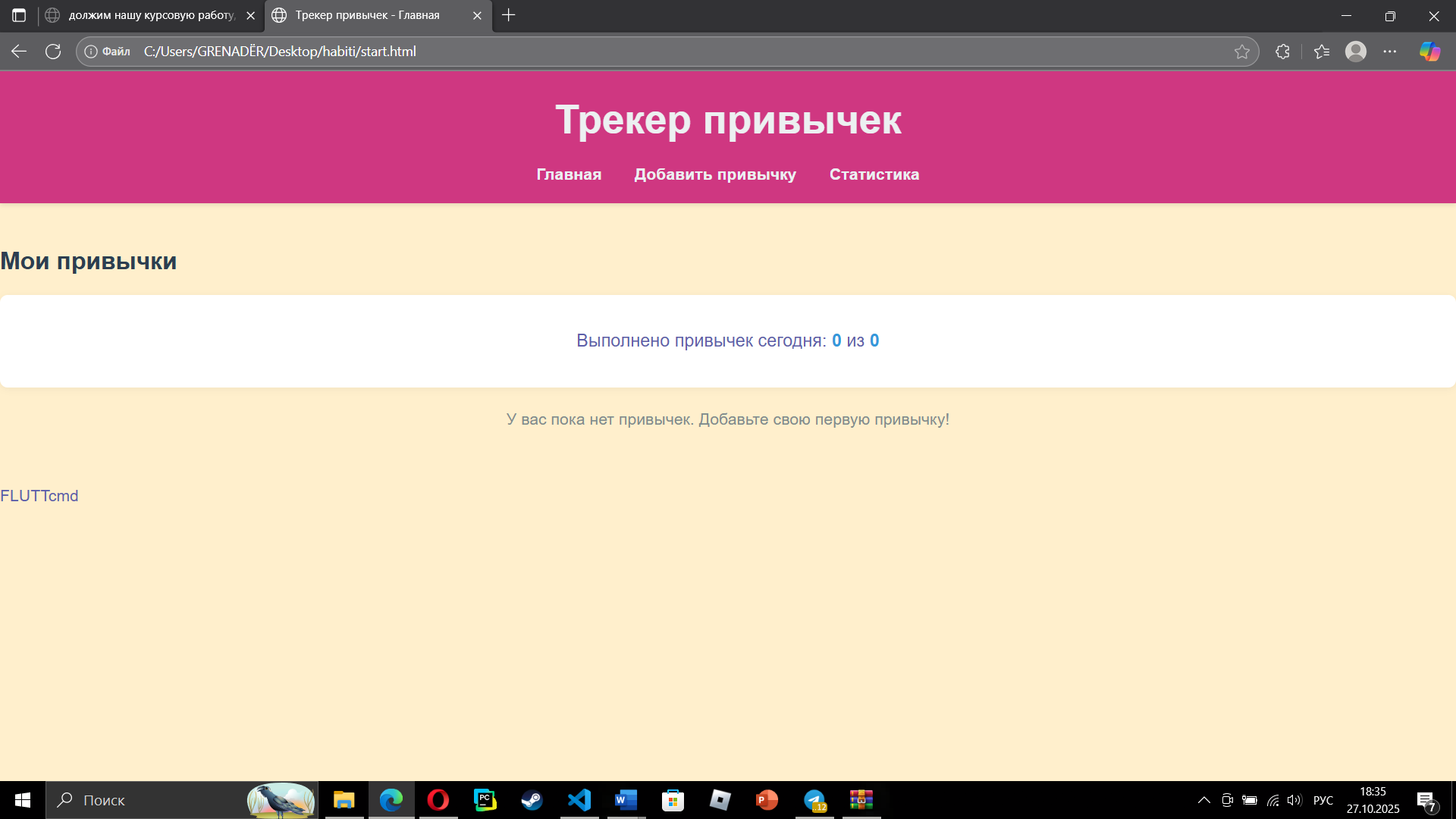
В рамках обсуждения перспектив развития, были обозначены необходимость и пути реализации серверной части (Backend) для масштабирования приложения, добавления многопользовательских функций, синхронизации данных и отправки напоминаний. Также были представлены идеи по расширенному функционалу, такому как геймификация, социальные элементы и интеграция с другими сервисами, а также важность углубленного тестирования и оптимизации производительности.

Проведенная работа позволила продемонстрировать комплексный подход к созданию современного веб-приложения, начиная с фундаментальных основ HTML и заканчивая обсуждением путей развития с использованием передовых технологий. Разработанный прототип "Трекера привычек" является рабочей основой, способной к дальнейшему совершенствованию и превращению в полнофункциональный инструмент для пользователей, стремящихся к саморазвитию и формированию полезных привычек.

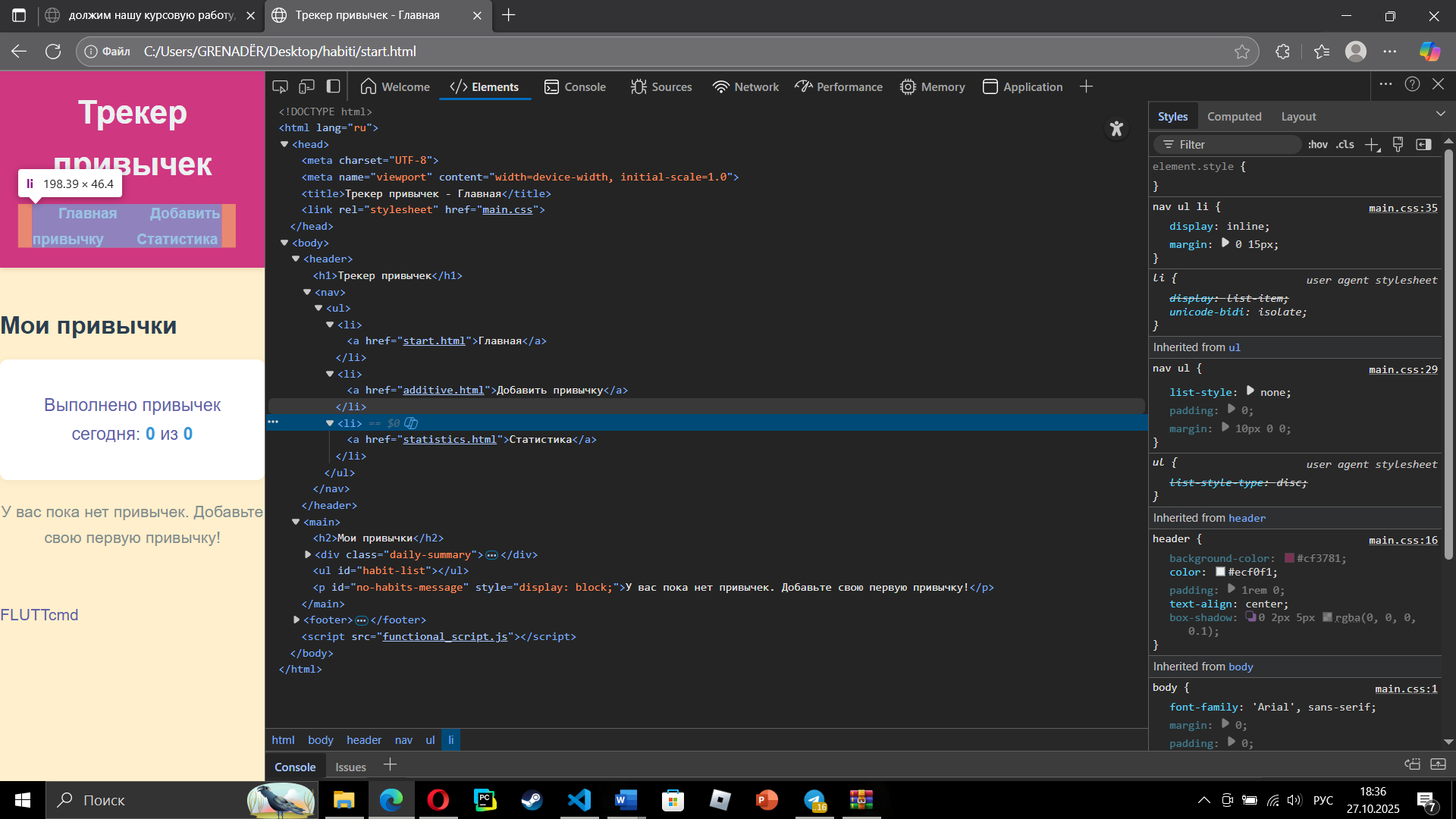
# Приложения



Вид добавления привычки



Вид главной страницы



Демонстрация функций li и ul

# Список использованной литературы

1. W3C (World Wide Web Consortium). «HTML Living Standard». <https://html.spec.whatwg.org/multipage/> (Дата обращения: 2 октября 2025 г.)

2. W3C (World Wide Web Consortium). «CSS Snapshot 2023». <https://www.w3.org/TR/css-2023/> (Дата обращения: 3 октября 2025г.)

3. Google Developers. «Web Fundamentals: Introduction to CSS». Получено из <https://developers.google.com/web/fundamentals/> ("CSS", "HTML", "JavaScript"). (Дата обращения: 8 октября 2025 г.)

4. CSS-Tricks. «Articles and Tutorials on CSS, Flexbox, Grid, and more». <https://css-tricks.com/> (Дата обращения: 18 октября 2025г.)

5. JavaScript.info. «Современный учебник JavaScript» <https://learn.javascript.ru/> (Дата обращения: 18 октября 2025г.)