Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Российский университет транспорта»

РУТ (МИИТ)

Институт управления и цифровых технологий

Кафедра «Цифровые технологии управления транспортными процессами»

ДИСЦИПЛИНА (МОДУЛЬ)

Разработка веб-приложений

Курсовой проект

Тема: Разработка веб-приложения: блог о космосе

Выполнил: Игнатьев В.Д.

Группа: УВПв-421

Проверил: Мельничук П.М.

Москва – 2025

Оглавление

[ВВЕДЕНИЕ 3](#_Toc200804022)

[ОПИСАНИЕ ПРОЕКТА 4](#_Toc200804023)

[Описание проекта 4](#_Toc200804024)

[ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ 5](#_Toc200804025)

[Структура HTML 5](#_Toc200804026)

[Общая структура документа 5](#_Toc200804027)

[Шапка страницы 7](#_Toc200804028)

[Основной контент 7](#_Toc200804029)

[Динамические посты 7](#_Toc200804030)

[Боковая панель: форма создания постов 9](#_Toc200804031)

[Боковая панель: архив 10](#_Toc200804032)

[Подключение скрипта 11](#_Toc200804033)

[Итоговая структура HTML-кода 11](#_Toc200804034)

[Структура CSS 13](#_Toc200804035)

[Глобальные стили 13](#_Toc200804036)

[Шапка 13](#_Toc200804037)

[Основной контент 14](#_Toc200804038)

[Посты 14](#_Toc200804039)

[Форма создания постов 15](#_Toc200804040)

[Архив 16](#_Toc200804041)

[Итоговая структура CSS-кода 17](#_Toc200804042)

[Структура JavaScript 21](#_Toc200804043)

[Функция инициализации приложения 21](#_Toc200804044)

[Функция addPost 22](#_Toc200804045)

[Анонимные функции удаления и архивации поста 24](#_Toc200804046)

[Функция обработки отправки формы 26](#_Toc200804047)

[Начальные посты 27](#_Toc200804048)

[Дизайн и пользовательский опыт 29](#_Toc200804049)

[Тестирование и Валидация 30](#_Toc200804050)

[Результат добавления поста 30](#_Toc200804051)

[Удаление и архивация 31](#_Toc200804052)

[Заключение 33](#_Toc200804053)

[Список литературы 34](#_Toc200804054)

[Приложения 35](#_Toc200804055)

[Приложение 1. HTML-код проекта 35](#_Toc200804056)

[Приложение 2. CSS-код проекта 37](#_Toc200804057)

[Приложение 3. JavaScript-код проекта 41](#_Toc200804058)

# ВВЕДЕНИЕ

В современном мире интерес к космосу и астрономии стремительно растет благодаря новым открытиям, миссиям на Марс и популяризации науки через медиа. Веб-приложения, такие как блоги, играют ключевую роль в распространении знаний, предоставляя платформу для обмена информацией и вовлечения широкой аудитории. Создание специализированного блога о космосе позволяет не только делиться научными фактами, но и вдохновлять пользователей на изучение Вселенной.

Целью данного курсового проекта является разработка веб-приложения — блога, посвященного космосу, с использованием базовых технологий веб-разработки: HTML для структуры, CSS для стилизации и JavaScript для интерактивности. Проект направлен на освоение основ веб-программирования и демонстрацию навыков создания функциональных веб-страниц без применения сторонних библиотек.

Задачи проекта:

1. Разработать структуру блога с учетом удобства использования.
2. Реализовать функциональность для создания, удаления и архивирования постов.
3. Оформить приложение в соответствии с заданными стилистическими требованиями.
4. Обеспечить корректную работу приложения в современных браузерах.

Актуальность проекта обусловлена необходимостью популяризации науки и развития навыков веб-разработки, которые востребованы в современной IT-индустрии. Блог о космосе может стать платформой для образовательного контента, привлекая как энтузиастов, так и профессионалов.

# ОПИСАНИЕ ПРОЕКТА

## Описание проекта

Блог о космосе — это веб-приложение, предназначенное для публикации и управления статьями на космическую тематику. Приложение состоит из трех основных компонентов:

1. Заголовок блога, отображающий название.
2. Раздел постов, где отображаются статьи пользователей.
3. Боковая панель с формой для создания новых постов и списком архива.

Каждый пост включает следующие элементы:

1. Автор поста.
2. Категория (например, "Астрономия", "Космические технологии").
3. Заголовок.
4. Содержимое.
5. Кнопки "Удалить" и "Архивировать".

Пользователи могут создавать новые посты, заполняя форму, удалять существующие посты или архивировать их, при этом заголовки архивированных постов добавляются в список архива. Приложение разработано с учетом простоты и интуитивности интерфейса, что делает его доступным для широкой аудитории.

# ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ

## Структура HTML

HTML-код веб-приложения "Космический блог" представляет собой основу одностраничного приложения, разработанного для создания, отображения и управления постами на космическую тематику. Он построен с использованием современных стандартов HTML5, включая семантические теги для улучшения структуры, доступности и читаемости. Ниже представлено детальное описание каждого тега, его атрибутов и назначения в контексте приложения. Описание включает как статические элементы, заданные непосредственно в HTML, так и те, что добавляются динамически через JavaScript, чтобы дать полное представление о структуре.

### Общая структура документа

1. <!DOCTYPE html> — это декларация типа документа, которая указывает браузеру, что страница использует HTML5. Она не является тегом, а специальной инструкцией, обязательной для современных веб-страниц. Без неё браузер может некорректно интерпретировать код, переключаясь в режим совместимости.
2. <html> — корневой элемент HTML-документа, который оборачивает весь контент страницы. Он задаёт начало и конец документа. В данном случае используется с атрибутом lang="ru", что означает, что основной язык страницы — русский. Это важно для поисковых систем и программ экранного чтения, улучшая доступность.
3. <head> — тег определяет раздел документа, содержащий метаинформацию, заголовок и подключение внешних ресурсов. Он не отображается напрямую на странице, но играет ключевую роль в настройке и поведении документа.
4. <meta charset="UTF-8"> — этот тег находится внутри <head> и устанавливает кодировку символов документа как UTF-8. UTF-8 поддерживает широкий набор символов, включая кириллицу, что необходимо для корректного отображения русского текста, такого как "Космический блог".
5. <meta name="viewport" content="width=device-width, initial-scale=1.0"> — также расположен в <head>. Этот тег отвечает за адаптивность страницы на мобильных устройствах. Атрибут width=device-width подстраивает ширину контента под ширину экрана устройства, а initial-scale=1.0 задаёт начальный масштаб в 1:1, обеспечивая правильное отображение без искажений.
6. <title> — тег внутри <head>, который задаёт заголовок страницы, отображаемый во вкладке браузера. В данном случае его содержимое — "Космический блог". Это помогает пользователю и поисковым системам понять тематику страницы.
7. <link> — тег в <head> подключает внешний файл стилей. С атрибутами rel="stylesheet" (указывает, что это таблица стилей) и href="styles.css" (путь к файлу styles.css), он позволяет применять CSS-правила для оформления страницы, таких как цвета, шрифты и расположение элементов.
8. <body> — основной тег, содержащий весь видимый контент страницы. Он объединяет шапку, основной контент, боковую панель и скрипты, обеспечивая структуру приложения, с которой взаимодействует пользователь.

### Шапка страницы

1. <header> — семантический тег внутри <body>, обозначающий верхнюю часть страницы (шапку). Он используется для размещения заголовка блога и, при необходимости, других элементов навигации. В данном случае содержит только заголовок.
2. <h1> — тег внутри <header>, представляющий заголовок первого уровня. Его содержимое — "Космический блог". Это главный заголовок страницы, который выделяет её тему и улучшает SEO, так как поисковые системы придают особое значение <h1>.

### Основной контент

1. <main> — семантический тег внутри <body>, который обозначает основной контент страницы, отличая его от шапки, подвала или боковых панелей. В "Космическом блоге" он включает раздел постов и боковую панель.
2. <section> (с id="posts") — тег внутри <main>, представляющий раздел для отображения постов. Атрибут id="posts" уникально идентифицирует этот элемент, позволяя JavaScript обращаться к нему для динамического добавления контента. Изначально этот раздел пуст, так как посты создаются и вставляются программно.
3. <aside> — семантический тег внутри <main>, обозначающий боковую панель. Он содержит дополнительные элементы интерфейса, такие как форма создания постов и архив, которые дополняют основной контент, но не являются его частью.

### Динамические посты

Динамические посты добавляются через JavaScript.

1. <article> — семантический тег, используемый для каждого отдельного поста внутри <section id="posts">. Он добавляется динамически через JavaScript и обозначает независимый блок контента (пост), который может существовать отдельно от других.
2. <h3> — тег внутри <article>, представляющий заголовок поста. Это заголовок третьего уровня, который используется для названия каждого поста, например, "Путешествие к Марсу". Он меньше по значимости, чем <h1>, но выделяет ключевую информацию.
3. <p> (для автора) — тег параграфа внутри <article>, отображающий имя автора поста, например, "Автор: Иван Иванов". Он повторяется для разных данных поста, но здесь выделен для автора.
4. <p> (для категории) — ещё один тег параграфа внутри <article>, показывающий категорию поста, например, "Категория: Планеты". Это помогает классифицировать контент.
5. <p> (для содержимого) — тег параграфа внутри <article>, содержащий основной текст поста, например, "Сегодня мы расскажем о Марсе...". Это главная часть поста, где размещается пользовательский контент.
6. <button> (с class="delete") — тег кнопки внутри <article> с классом class="delete". Содержимое кнопки — "Удалить". Она позволяет пользователю удалить пост, а класс используется для стилизации в CSS и обработки события в JavaScript.
7. <button> (с class="archive") — тег кнопки внутри <article> с классом class="archive". Содержимое — "Архивировать". Эта кнопка отправляет пост в архив, а класс служит для стилизации и функциональности.

### Боковая панель: форма создания постов

1. <section> (с id="create-post") — тег внутри <aside>, обозначающий секцию для создания нового поста. Атрибут id="create-post" позволяет обращаться к этому блоку через JavaScript и CSS.
2. <h2> (в <section id="create-post">) — тег заголовка второго уровня внутри <section id="create-post">. Его содержимое — "Создать пост". Он обозначает назначение секции и визуально отделяет её от других частей.
3. <form> (с id="post-form") — тег формы внутри <section id="create-post">. Атрибут id="post-form" используется для идентификации формы в JavaScript. Она собирает данные для нового поста и отправляет их для обработки.
4. <label> (для поля "Автор") — тег метки внутри <form> с атрибутом for="author". Содержимое — "Автор:". Он связан с полем ввода <input id="author"> и улучшает доступность, указывая, что вводится.
5. <input> (с type="text" id="author" name="author") — тег поля ввода внутри <form>. Атрибуты: type="text" (текстовое поле), id="author" (уникальный идентификатор), name="author" (имя для отправки данных). Пользователь вводит сюда имя автора.
6. <label> (для поля "Категория") — тег метки внутри <form> с атрибутом for="category". Содержимое — "Категория:". Связан с <input id="category">.
7. <input> (с type="text" id="category" name="category") — тег поля ввода внутри <form>. Атрибуты: type="text", id="category", name="category". Используется для ввода категории поста.
8. <label> (для поля "Заголовок") — тег метки внутри <form> с атрибутом for="title". Содержимое — "Заголовок:". Связан с <input id="title">.
9. <input> (с type="text" id="title" name="title") — тег поля ввода внутри <form>. Атрибуты: type="text", id="title", name="title". Сюда вводится заголовок поста.
10. <label> (для поля "Содержимое") — тег метки внутри <form> с атрибутом for="content". Содержимое — "Содержимое:". Связан с <textarea id="content">.
11. <textarea> (с id="content" name="content") — тег текстового поля внутри <form>. Атрибуты: id="content", name="content". Это многострочное поле для ввода содержимого поста, например, текста о космосе.
12. <button> (с type="submit") — тег кнопки внутри <form>. Атрибут type="submit" указывает, что она отправляет форму. Содержимое — "Создать". При нажатии данные формы обрабатываются JavaScript для создания нового поста.

### Боковая панель: архив

1. <section> (с id="archive") — тег внутри <aside>, обозначающий секцию архива. Атрибут id="archive" используется для обращения к этому блоку.
2. <h2> (в <section id="archive">) — тег заголовка второго уровня внутри <section id="archive">. Содержимое — "Архив". Указывает назначение секции.
3. <ul> (с id="archive-list") — тег ненумерованного списка внутри <section id="archive">. Атрибут id="archive-list" позволяет добавлять элементы списка через JavaScript. Изначально пуст.
4. <li> (в <ul id="archive-list">) — тег элемента списка внутри <ul>. Добавляется динамически для каждого архивированного поста, содержит его заголовок, например, "Путешествие к Марсу".

### Подключение скрипта

<script> — тег в конце <body> с атрибутом src="script.js". Подключает внешний файл JavaScript, который отвечает за интерактивность: обработку формы, добавление постов, удаление и архивирование.

### Итоговая структура HTML-кода

Этот код создаёт прочную основу для интерактивного блога, где пользователи могут добавлять, удалять и архивировать посты, а семантическая разметка обеспечивает доступность и удобство разработки.

На рисунке 1 представлена структура HTML кода.



Рисунок 1. HTML-код приложения.

## Структура CSS

CSS создаёт современный дизайн с тёмной шапкой, гибкой компоновкой (3:1 для контента и боковой панели), зелёными акцентами и пунктирными границами постов. Тени и скруглённые углы добавляют объём, а кнопки меняют цвет при фокусе для удобства. Всё это делает "Космический блог" стильным и удобным.

### Глобальные стили

1. body
   1. font-family: Arial, sans-serif; — устанавливает Arial как основной шрифт с запасным вариантом sans-serif для единообразия текста на всех устройствах.
   2. margin: 0; — убирает внешние отступы страницы, чтобы контент занимал всё пространство.
   3. padding: 0; — убирает внутренние отступы <body>, предотвращая лишние зазоры.

### Шапка

1. header
   1. background: #333; — задаёт тёмно-серый фон (#333 — шестнадцатеричный код цвета), выделяя шапку.
   2. color: white; — устанавливает белый цвет текста для контраста с фоном.
   3. padding: 1em; — добавляет внутренний отступ 1em (относительно размера шрифта) для пространства вокруг текста.
   4. text-align: center; — центрирует содержимое шапки, делая заголовок симметричным.

### Основной контент

1. main
   1. display: flex; — включает flexbox, позволяя располагать раздел постов и боковую панель горизонтально.
   2. padding: 1em; — добавляет отступ 1em вокруг содержимого для "воздуха" между краями и элементами.
2. #posts
   1. flex: 3; — задаёт разделу постов ширину в 3 части flex-контейнера, делая его шире боковой панели.
   2. margin-right: 1em; — добавляет отступ справа 1em, отделяя посты от боковой панели.
3. aside
   1. flex: 1; — задаёт боковой панели ширину в 1 часть flex-контейнера, делая её уже постов.
   2. background: #f4f4f4; — устанавливает светло-серый фон для визуального отличия.
   3. padding: 1em; — добавляет внутренний отступ 1em для содержимого (формы и архива).

### Посты

1. article
   1. border: dashed 0.1rem #006b09; — создаёт пунктирную зелёную границу толщиной 0.1rem (#006b09 — тёмно-зелёный), выделяя каждый пост.
   2. border-radius: 15px; — скругляет углы на 15px для мягкого, современного вида.
   3. box-shadow: 0.5em 1em 0.5em 0em rgba(64, 73, 59, 0.281); — добавляет тень: 0.5em смещение по горизонтали, 1em по вертикали, 0.5em размытие, цвет с прозрачностью 0.281 для эффекта глубины.
   4. padding: 1em; — задаёт внутренний отступ 1em, чтобы содержимое не прилегало к границам.
   5. margin-bottom: 1em; — добавляет отступ снизу 1em для разделения постов.
2. article button.delete
   1. background: #cf3a15; — задаёт красный фон (#cf3a15), сигнализируя об удалении.
   2. color: white; — белый текст для читаемости.
   3. padding: 0.5em 1em; — отступы: 0.5em по вертикали, 1em по горизонтали, делая кнопку удобной для нажатия.
   4. border: none; — убирает стандартную границу кнопки.
   5. border-radius: 5px; — скругляет углы на 5px.
   6. cursor: pointer; — меняет курсор на указатель при наведении, показывая кликабельность.
3. article button.archive
   1. background: #A3AE9E; — задаёт серо-зелёный фон (#A3AE9E), отличая кнопку от "Удалить".
   2. Остальные свойства (color, padding, border, border-radius, cursor) идентичны кнопке .delete.

### Форма создания постов

1. #create-post h2
   1. color: black; — чёрный цвет текста заголовка "Создать пост".
   2. border-bottom: 1px solid #07b148; — зелёная нижняя граница (#07b148) толщиной 1px для отделения заголовка.
2. #create-post label
   1. display: block; — делает метки блочными, размещая их вертикально.
   2. margin-top: 1em; — отступ сверху 1em для пространства между метками.
3. #create-post input, #create-post textarea
   1. width: 100%; — ширина 100% для заполнения всей доступной области.
   2. border-bottom: 1px solid #40493B; — серая нижняя граница (#40493B) толщиной 1px для минималистичного вида.
   3. padding: 0.5em; — внутренний отступ 0.5em для удобства ввода.
4. #create-post button
   1. background: #006b09; — тёмно-зелёный фон (#006b09) для кнопки "Создать".
   2. color: white; — белый текст.
   3. border-radius: 5px; — скругление углов на 5px.
   4. padding: 0.5em 1em; — отступы как у кнопок постов.
   5. border: none; — без границы.
5. #create-post button:focus
   1. background: #07b148; — светло-зелёный фон (#07b148) при фокусе для визуальной обратной связи.

### Архив

1. #archive h2
   1. color: black; — чёрный текст заголовка "Архив".
   2. border-bottom: 1px solid #07b148; — зелёная нижняя граница, как в форме.
2. #archive ul
   1. list-style-type: none; — убирает маркеры списка для чистого вида.
3. #archive li
   1. margin-bottom: 0.5em; — отступ снизу 0.5em для разделения элементов списка.

### Итоговая структура CSS-кода

CSS-код "Космического блога" формирует чистый и функциональный интерфейс. Flexbox обеспечивает гибкую компоновку, зелёные акценты (границы, кнопки) и тёмный фон шапки создают космическую атмосферу. Свойства вроде border-radius и box-shadow добавляют современности, а padding и margin — удобства восприятия. Кнопки с cursor: pointer и фокус-стили улучшают интерактивность.

На рисунках 2-4 изображен CSS код приложения.

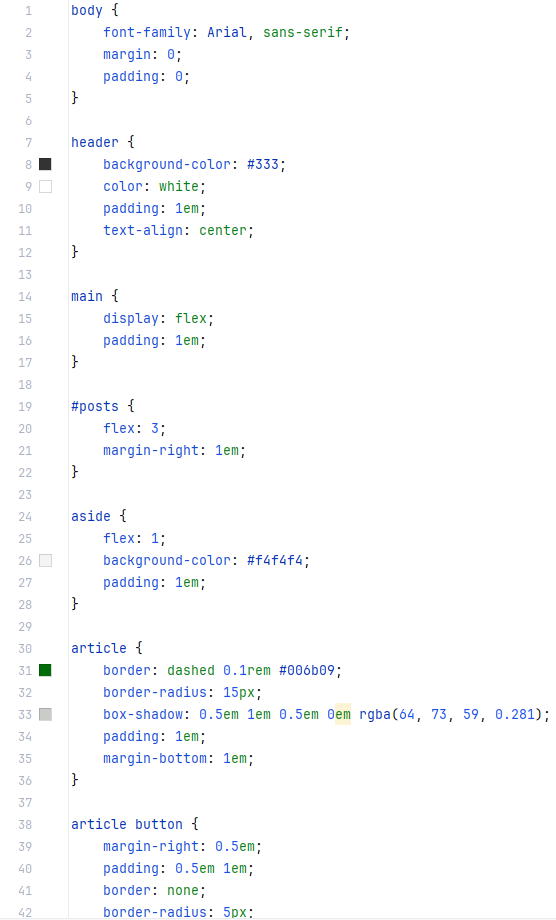


Рисунок 2. Первая часть структуры CSS-кода.

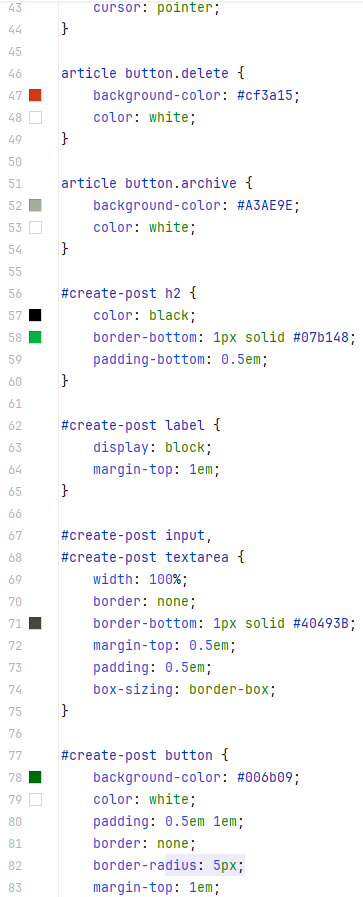


Рисунок 3. Вторая часть структуры CSS-кода

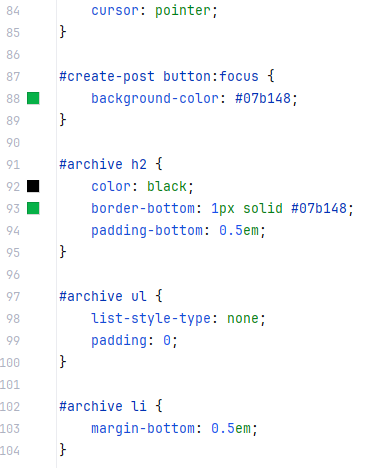


Рисунок 4. Третья часть структуры CSS-кода

## Структура JavaScript

Интерактивность реализована в файле script.js. Основные функции:

1. Создание постов: обработка формы, проверка заполненности полей, динамическое добавление постов.
2. Удаление постов: удаление элемента <article> из DOM.
3. Архивирование: перенос заголовка поста в список архива и удаление поста.

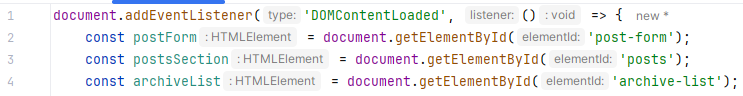
### Функция инициализации приложения

Эта функция запускает приложение, когда страница полностью загружается в браузере. Она обеспечивает, чтобы весь JavaScript-код начал работать только после того, как все элементы HTML (форма, кнопки, разделы) станут доступны.

Внутри обработчика выполняются все остальные действия: получение элементов страницы, вызов функции создания начальных постов и настройка обработки формы.

Без этой функции JavaScript мог бы попытаться работать с элементами страницы до их загрузки, что вызвало бы ошибки.

На рисунке 5 изображён код данной функции. Также на нём видно создание передача в переменную таких элементов, как форма для создания поста, секции для постов и список архивированных постов. JavaScript получает эти объекты через id, указанный в HTML (используется метод document.getElementById().

Рисунок 5. Код функции инициализации.

Переменные postForm, postsSection и archiveList содержат ссылки на соответствующие элементы DOM, готовые для использования в других функциях.

### Функция addPost

Эта функция создаёт новый пост на странице, отображая его в разделе постов с указанными данными (заголовок, автор, категория, содержимое) и добавляя кнопки для управления.

Входные параметры:

1. title (строка): Заголовок поста, например, "Исследование Млечного Пути".
2. author (строка): Имя автора, например, "Астро Фанат".
3. category (строка): Категория поста, например, "Астрономия".
4. content (строка): Текст поста, например, "Млечный Путь — наша родная галактика...".

Логика работы:

1. Создаёт новый элемент <article> с помощью document.createElement('article'), который будет контейнером для поста.
2. Задаёт содержимое элемента через innerHTML, формируя структуру поста:
3. <h3> для заголовка (title).
4. Три тега <p> для автора, категории и содержимого.
5. Две кнопки <button> с классами delete и archive для удаления и архивирования.
6. Находит кнопки внутри созданного поста с помощью article.querySelector('.delete') и article.querySelector('.archive').
7. Добавляет обработчики событий для кнопок:
8. Для кнопки "Удалить" вызывает функцию удаления поста.
9. Для кнопки "Архивировать" вызывает функцию архивирования.
10. Добавляет пост в раздел <section id="posts"> с помощью postsSection.appendChild(article).

В результате новый пост появляется на странице в разделе постов, с заголовком, автором, категорией, текстом и двумя кнопками, готовыми к взаимодействию.

Это основная функция для добавления контента. Она отвечает за то, чтобы новые посты, созданные пользователем или автоматически, отображались корректно и были интерактивными.

На рисунке 6. Изображён код вышеописанной функции.

Рисунок 6. Функция добавления поста.

### Анонимные функции удаления и архивации поста

Функция deleteBtn удаляет пост со страницы, когда пользователь нажимает кнопку "Удалить".

Функция archiveBtn переносит заголовок поста в список архива и удаляет пост из основного раздела, когда пользователь нажимает "Архивировать".

На вход не принимают параметров, так как работают с элементом <article>, к которому привязана кнопка.

Логика работы для кнопки удаления:

1. Реализована как анонимная функция внутри обработчика события deleteBtn.addEventListener('click', () => {...}).
2. Вызывает метод article.remove(), который удаляет весь элемент <article> (пост) из DOM (структуры страницы).
3. Удаление происходит мгновенно, без дополнительных проверок, так как это простое действие.

Логика работы для кнопки архивации:

1. Реализована как анонимная функция внутри archiveBtn.addEventListener('click', () => {...}).
2. Создаёт новый элемент <li> с помощью document.createElement('li').
3. Устанавливает текст элемента <li> равным заголовку поста (li.textContent = title).
4. Добавляет элемент <li> в список архива с помощью archiveList.appendChild(li).
5. Удаляет пост из раздела постов с помощью article.remove().

В результате работе функции удаления исчезает из раздела постов. Функция делает тоже самое, но заголовок поста появляется в списке архива. Это позволяет сохранять историю постов, перемещая их заголовки в архив вместо полного удаления.

На рисунке 7 изображены обе функции.

Рисунок 7. Код функций удаления и архивации постов.

### Функция обработки отправки формы

Эта функция обрабатывает данные, введённые пользователем в форму, создаёт новый пост и управляет поведением формы. Принимает объект события e (событие отправки формы), автоматически передаваемый браузером.

Логика работы:

1. Реализована как анонимная функция внутри postForm.addEventListener('submit', (e) => {...}).
2. Вызывает e.preventDefault(), чтобы предотвратить стандартное поведение формы (перезагрузку страницы).
3. Извлекает данные из полей формы с помощью document.getElementById().value:
   1. author — имя автора из <input id="author">.
   2. category — категория из <input id="category">.
   3. title — заголовок из <input id="title">.
   4. content — содержимое из <textarea id="content">.
4. Проверяет, заполнены ли все поля с помощью условия if (author && category && title && content).
5. Если все поля заполнены:
   1. Вызывает addPost(title, author, category, content) для создания поста.
   2. Очищает форму с помощью postForm.reset().
6. Если хотя бы одно поле пустое, показывает сообщение alert('Пожалуйста, заполните все поля.').

Результат: если данные валидны, на странице появляется новый пост, а форма очищается. Если данные неполные, пользователь видит предупреждение.

Это основная функция для взаимодействия пользователя с блогом. Она позволяет добавлять новый контент, обеспечивая удобство и валидацию.

На рисунке 8 представлен код данной функции.

Рисунок 8. Код функции добавления поста.

### Начальные посты

Также в коде (см. рисунок 9) есть создание начальных постов для демонстрации, которые отображаются на странице после перезагрузки.

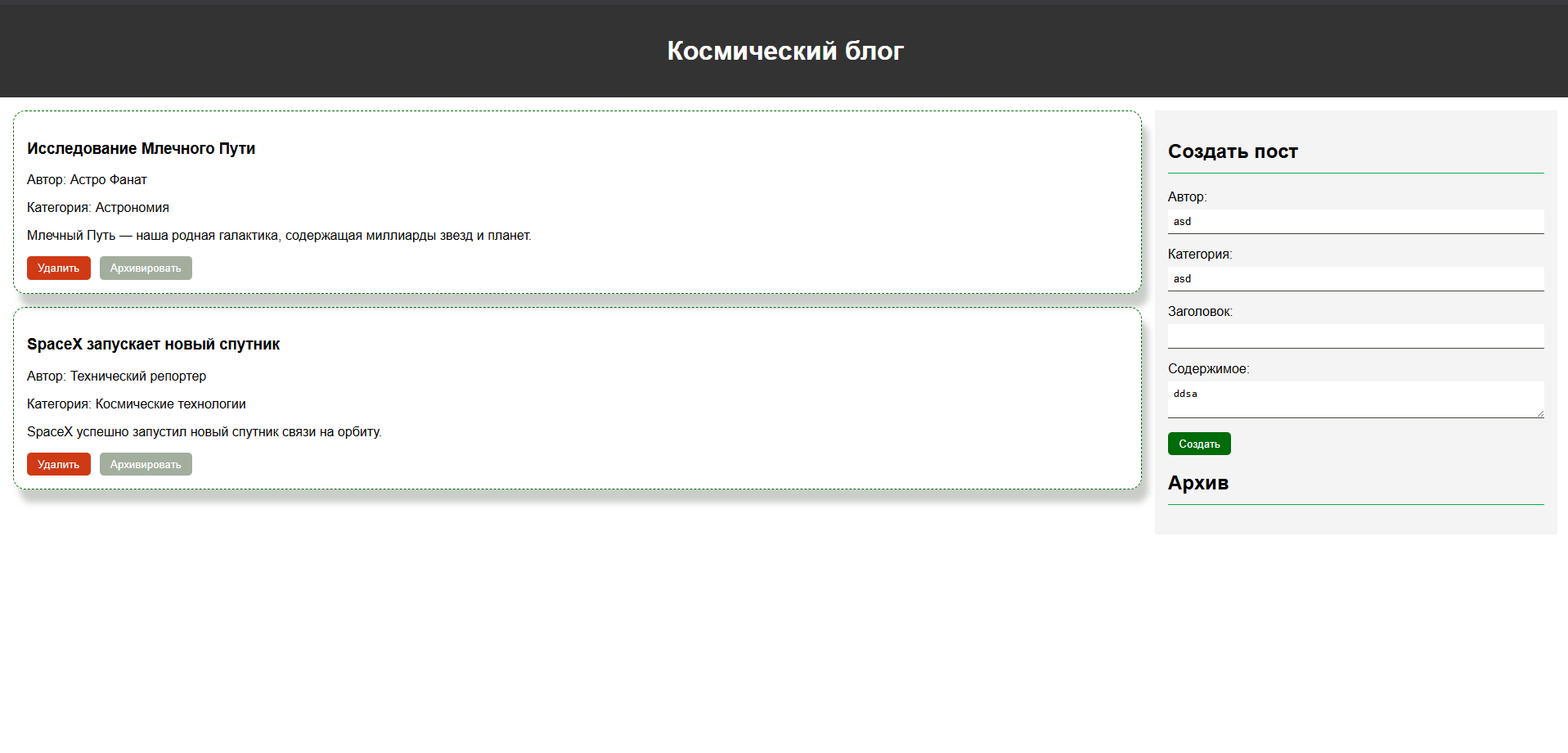
Рисунок 9. Код добавление начальных постов.

# Дизайн и пользовательский опыт

Дизайн блога разработан с учетом заданных требований и тематики космоса. Цветовая палитра включает зеленые оттенки (#006b09, #07b148), которые могут ассоциироваться с научной фантастикой или внеземными пейзажами. Макет использует flexbox для разделения контента на основной раздел (75%) и боковую панель (25%), обеспечивая удобство восприятия.

Интерфейс интуитивно понятен: форма создания постов проста в использовании, а кнопки управления постами четко обозначены. Элементы управления имеют визуальную обратную связь (например, изменение цвета кнопки при фокусе), что улучшает пользовательский опыт.

На рисунке 10 изображён финальный дизайн проекта.

Рисунок 10. Финальный дизайн проекта.

# Тестирование и Валидация

Приложение протестировано в браузерах Chrome, Firefox и Edge. Проверены следующие аспекты:

1. Корректность отображения макета на разных разрешениях экрана.
2. Работоспособность формы создания постов.
3. Функциональность кнопок "Удалить" и "Архивировать".
4. Валидация HTML и CSS через онлайн-сервисы.

## Результат добавления поста

Для тестирования добавления была заполнена форма (см. рисунок 11).

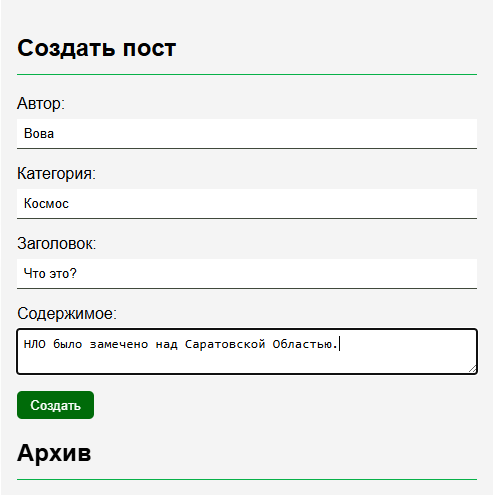


Рисунок 11. Заполненная форма добавления поста.

После нажатия кнопки "Создать", в основном блоке появляется пост со всеми полями (см. рисунок 12).

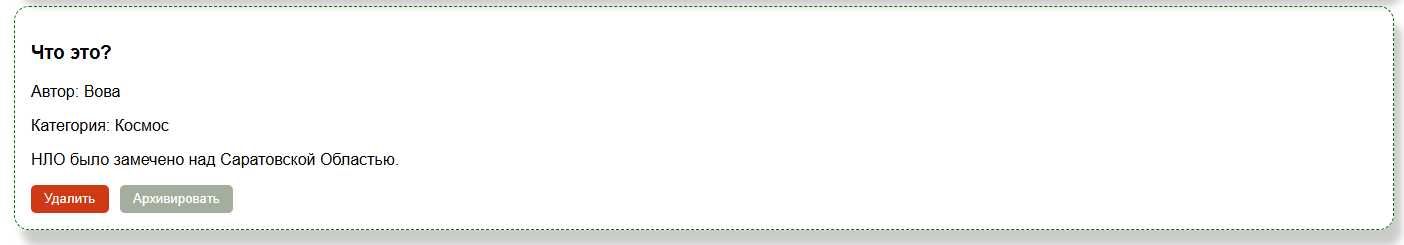


Рисунок 12. Созданный пост.

При попытке добавить пост, не заполнив все поля, появляется уведомление с просьбой заполнить все поля. На рисунке 13 изображено это уведомление.

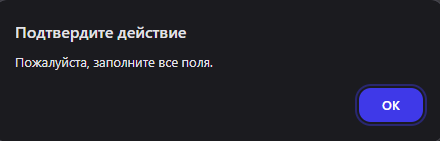
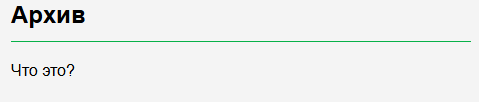


Рисунок 13. Уведомление с просьбой заполнить все поля формы.

## Удаление и архивация

После нажатия кнопки "Удалить" пост исчезает без следа. После нажатия кнопки "Архивировать" пост исчезает, а заголовок поста добавляется в архив (см. рисунок 14).

Рисунок 14. Пост в архиве.

Ошибок в работе приложения не выявлено.

# Заключение

В рамках курсового проекта разработано веб-приложение — блог о космосе, соответствующее всем заданным требованиям. Проект позволил освоить основы веб-разработки, включая работу с HTML, CSS и JavaScript, а также научиться структурировать проектную документацию.

Дальнейшее развитие приложения может включать:

1. Интеграцию с API для получения новостей о космосе.
2. Добавление системы комментариев.
3. Реализацию локального хранения данных с помощью localStorage.

# Список литературы

1. Иванов А.А. Основы веб-разработки: HTML, CSS, JavaScript. — М.: Техника, 2023. — 350 с.
2. Петрова Б.В. Дизайн веб-интерфейсов. — СПб.: Питер, 2022. — 280 с.
3. Сидоров В.Г. JavaScript для начинающих. — М.: ДМК Пресс, 2024. — 400 с.
4. Космические исследования: современные достижения // Научный журнал. — 2025. — №5. — С. 10–25.
5. Space.com [Электронный ресурс]. — URL: https://www.space.com (дата обра- щения: 13.06.2025).
6. NASA официальный сайт [Электронный ресурс]. — URL: https://www.nasa.gov (дата обращения: 13.06.2025).
7. Карпов Д.Е. Астрономия для всех. — М.: АСТ, 2021. — 200 с.
8. Web Design Trends 2025 [Электронный ресурс]. — URL: https://www.webdesigntren (дата обращения: 13.06.2025).

# Приложения

## Приложение 1. HTML-код проекта

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Космический блог</title>  
 <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
<header>  
 <h1>Космический блог</h1>  
</header>  
<main>  
 <section id="posts">  
 *<!-- посты -->* </section>  
 <aside>  
 <section id="create-post">  
 <h2>Создать пост</h2>  
 <form id="post-form">  
 <label for="author">Автор:</label>  
 <input type="text" id="author" name="author">  
 <label for="category">Категория:</label>  
 <input type="text" id="category" name="category">  
 <label for="title">Заголовок:</label>  
 <input type="text" id="title" name="title">  
 <label for="content">Содержимое:</label>  
 <textarea id="content" name="content"></textarea>  
 <button type="submit">Создать</button>  
 </form>  
 </section>  
 <section id="archive">  
 <h2>Архив</h2>  
 <ul id="archive-list">  
 *<!-- архив -->* </ul>  
 </section>  
 </aside>  
</main>  
<script src="script.js"></script>  
</body>  
</html>

## Приложение 2. CSS-код проекта

body {  
 font-family: Arial, sans-serif;  
 margin: 0;  
 padding: 0;  
}  
  
header {  
 background-color: #333;  
 color: white;  
 padding: 1em;  
 text-align: center;  
}  
  
main {  
 display: flex;  
 padding: 1em;  
}  
  
#posts {  
 flex: 3;  
 margin-right: 1em;  
}  
  
aside {  
 flex: 1;  
 background-color: #f4f4f4;  
 padding: 1em;  
}  
  
article {  
 border: dashed 0.1rem #006b09;  
 border-radius: 15px;  
 box-shadow: 0.5em 1em 0.5em 0em rgba(64, 73, 59, 0.281);  
 padding: 1em;  
 margin-bottom: 1em;  
}  
  
article button {  
 margin-right: 0.5em;  
 padding: 0.5em 1em;  
 border: none;  
 border-radius: 5px;  
 cursor: pointer;  
}  
  
article button.delete {  
 background-color: #cf3a15;  
 color: white;  
}  
  
article button.archive {  
 background-color: #A3AE9E;  
 color: white;  
}  
  
#create-post h2 {  
 color: black;  
 border-bottom: 1px solid #07b148;  
 padding-bottom: 0.5em;  
}  
  
#create-post label {  
 display: block;  
 margin-top: 1em;  
}  
  
#create-post input,  
#create-post textarea {  
 width: 100%;  
 border: none;  
 border-bottom: 1px solid #40493B;  
 margin-top: 0.5em;  
 padding: 0.5em;  
 box-sizing: border-box;  
}  
  
#create-post button {  
 background-color: #006b09;  
 color: white;  
 padding: 0.5em 1em;  
 border: none;  
 border-radius: 5px;  
 margin-top: 1em;  
 cursor: pointer;  
}  
  
#create-post button:focus {  
 background-color: #07b148;  
}  
  
#archive h2 {  
 color: black;  
 border-bottom: 1px solid #07b148;  
 padding-bottom: 0.5em;  
}  
  
#archive ul {  
 list-style-type: none;  
 padding: 0;  
}  
  
#archive li {  
 margin-bottom: 0.5em;  
}

## Приложение 3. JavaScript-код проекта

document.addEventListener('DOMContentLoaded', () => {  
 const postForm = document.getElementById('post-form');  
 const postsSection = document.getElementById('posts');  
 const archiveList = document.getElementById('archive-list');  
  
 function addPost(title, author, category, content) {  
 const article = document.createElement('article');  
 article.innerHTML = `  
 <h3>${title}</h3>  
 <p>Автор: ${author}</p>  
 <p>Категория: ${category}</p>  
 <p>${content}</p>  
 <button class="delete">Удалить</button>  
 <button class="archive">Архивировать</button>  
 `;  
  
 const deleteBtn = article.querySelector('.delete');  
 const archiveBtn = article.querySelector('.archive');  
  
 deleteBtn.addEventListener('click', () => {  
 article.remove();  
 });  
  
 archiveBtn.addEventListener('click', () => {  
 const li = document.createElement('li');  
 li.textContent = title;  
 archiveList.appendChild(li);  
 article.remove();  
 });  
  
 postsSection.appendChild(article);  
 }  
  
 *// начальные посты* addPost("Исследование Млечного Пути", "Астро Фанат", "Астрономия", "Млечный Путь — наша родная галактика, содержащая миллиарды звезд и планет.");  
 addPost("SpaceX запускает новый спутник", "Технический репортер", "Космические технологии", "SpaceX успешно запустил новый спутник связи на орбиту.");  
  
 postForm.addEventListener('submit', (e) => {  
 e.preventDefault();  
 const author = document.getElementById('author').value;  
 const category = document.getElementById('category').value;  
 const title = document.getElementById('title').value;  
 const content = document.getElementById('content').value;  
  
 if (author && category && title && content) {  
 addPost(title, author, category, content);  
 postForm.reset();  
 } else {  
 *alert*('Пожалуйста, заполните все поля.');  
 }  
 });  
});