МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Компьютерные языки разметки»

Тема: «Веб-сайт «Интернет-магазин одежды»»

**Исполнитель**

студент 1 курса 3 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Демидович П.А.

подпись, дата

**Руководитель**

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Харланович А.В

должность, учен. степень, ученое звание подпись, дата

Допущена к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ А.В. Харланович

подпись дата инициалы и фамилия

Минск, 2024

**Содержание**

[Введение 3](#_Toc134374022)

[1. Постановка задачи 5](#_Toc134374023)

[1.1 Обзор аналогичных решений 5](#_Toc134374024)

[1.2 Техническое задание 6](#_Toc134374025)

[1.3 Выбор средств реализации программного продукта 7](#_Toc134374026)

[1.4 Вывод 8](#_Toc134374027)

[2. Проектирование страниц веб-сайта 9](#_Toc134374028)

[2.1 Выбор способа вёрстки 9](#_Toc134374029)

[2.2 Выбор стилевого оформления 9](#_Toc134374030)

[2.3 Выбор шрифтового оформления 9](#_Toc134374031)

[2.4 Разработка логотипа 10](#_Toc134374032)

[2.5 Разработка пользовательских элементов 10](#_Toc134374033)

[2.6 Разработка спецэффектов. 10](#_Toc134374034)

[2.7 Вывод. 12](#_Toc134374035)

[3. Реализация структуры веб-сайта 13](#_Toc134374036)

[3.1 Структура HTML-документа 13](#_Toc134374037)

[3.2 Добавление таблиц стилей CSS 15](#_Toc134374038)

[3.3 Использование стандартов XML (SVG) 17](#_Toc134374039)

[3.4 Управление элементами DOM 18](#_Toc134374040)

[3.5 Вывод 22](#_Toc134374041)

[4 Тестирование веб-сайта 23](#_Toc134374042)

[4.1 Адаптивный дизайн веб-сайта 23](#_Toc134374043)

[4.2 Кроссбраузерность веб-сайта 25](#_Toc134374044)

[4.3 Руководство пользователя 27](#_Toc134374045)

[4.4 Тестирование кода 27](#_Toc134374046)

[4.5 Вывод 28](#_Toc134374046)

[Заключение 30](#_Toc134374047)

[Использованные литературные источники 31](#_Toc134374048)

[Приложение А Макет страниц 32](#_Toc134374050)

[Приложение Б Листинг HTML-документа 38](#_Toc134374051)

[Приложение В Листинг CSS 39](#_Toc134374052)

[Приложение Г Листинг SVG 45](#_Toc134374054)

[Приложение Д Листинг JavaScrpit 45](#_Toc134374055)

**Введение**

Интернет-магазин одежды — это электронная платформа, где можно приобрести различные виды одежды и аксессуары через интернет. Он предлагает широкий ассортимент товаров, включая мужскую, женскую и детскую одежду, обувь, сумки, украшения и многое другое. В интернет-магазине одежды вы можете выбрать и заказать товары, просмотреть фотографии, описания и цены, а также сделать оплату онлайн. После оформления заказа товары доставляются вам по указанному адресу. Интернет-магазины одежды предлагают удобство покупок, широкий выбор товаров и различные преимущества, о которых мы уже говорили.

Существует множество причин, по которым создаются сайты интернет-магазинов одежды. Вот некоторые из них:

1. Удобство: Возможность покупать одежду прямо из дома или офиса, без необходимости посещать физический магазин. Это экономит время и усталость от походов по магазинам.

2. Широкий выбор: Интернет-магазины одежды предлагают огромный ассортимент товаров. Вы можете найти любой стиль, размер или цвет, который вам нужен, в одном месте.

3. Удобный поиск: С помощью фильтров и поисковых функций вы легко можете найти нужную вам одежду. Вы можете сузить поиск по цене, размеру, цвету и другим параметрам, что делает процесс выбора быстрым и удобным.

4. Отзывы и рейтинги: Большинство интернет-магазинов предоставляют возможность оставить отзывы и оценки для товаров. Это помогает вам принять информированное решение о покупке, основываясь на опыте других покупателей.

5. Скидки и акции: Интернет-магазины часто предлагают специальные скидки и акции, которые помогают сэкономить деньги на покупках. Вы можете легко найти лучшие предложения и сэкономить немало средств.

6. Доставка: Большинство интернет-магазинов предлагают доставку прямо к вашему дверному проему. Вы можете заказать одежду и получить ее в удобное для вас время, без необходимости тратить время на походы в магазины.

7. Возможность сравнения: В интернете легко сравнивать цены и предложения разных магазинов. Вы можете найти лучшую цену на нужную вам одежду, а также сравнить качество и условия доставки.

8. Глобальный доступ: Интернет-магазины позволяют вам покупать одежду из любой точки мира. Вы можете исследовать и заказывать модные тренды и уникальные предметы из разных стран, которые могут быть недоступны в вашем регионе.

9. Удобные способы оплаты: Интернет-магазины предлагают различные способы оплаты, включая кредитные карты, электронные платежные системы и даже наличные при доставке. Выберите наиболее удобный способ для вас.

10. Простой возврат и обмен товаров: Если вам не подходит размер или стиль одежды, интернет-магазины обычно предоставляют возможность возврата или обмена товаров. Это удобно и экономит ваше время.

11. Анонимность: В интернет-магазинах вы можете покупать одежду, сохраняя свою анонимность. Нет необходимости стесняться или ощущать дискомфорт в присутствии других покупателей.

12. Уникальные коллекции и дизайнерские бренды: Многие интернет-магазины предлагают эксклюзивные коллекции и предметы от известных дизайнеров. Вы можете найти уникальные и стильные вещи, которые помогут выделиться из толпы.

13. Подробные описания и фотографии товаров: В интернет-магазинах вы можете найти подробные описания и качественные фотографии товаров. Это поможет вам лучше понять, как выглядит и описывается одежда, перед тем как сделать покупку.

14. Персонализированные рекомендации: Многие интернет-магазины предлагают персонализированные рекомендации на основе ваших предпочтений и истории покупок. Это помогает вам находить новые вещи, которые соответствуют вашему стилю и вкусу.

15. Экологичность: Покупка одежды в интернет-магазинах может быть более экологичной, поскольку это позволяет избежать поездок на автомобиле или использования пластиковых упаковок в физических магазинах.

В целом, создание сайта интернет-магазина является актуальной темой для курсового проекта, которая позволяет не только улучшить навыки веб-разработки, но также может быть полезной как для личных, так и для коммерческих целей.

В процессе работы над проектом были поставлены следующие задачи:

* создание кросс-браузерного веб-сайта;
* разработка адаптивности для веб-страниц;
* разработка макета веб-сайта;
* тестирование веб-сайта;
* добавление анимации.

Основные цели курсового проекта: анализ статистики сайтов конкурентов, выявление положительных и отрицательных характеристик сайтов схожей тематики, улучшение навыков верстки и получение опыта разработки проектов, ознакомление с новыми технологиями для создания сайтов. Выполнение поставленных задач.

**1.** **Постановка задачи**

Цель общеобразовательного ботанического сайта - повышение общего уровня знаний пользователей о растительном мире и его значении для нашей жизни. Общеобразовательный ботанический сайт будет полезен для всех людей, интересующихся природой.

**1.1 Обзор аналогичных решений**

При создании сайта интернет-магазина важно использовать различные аналитические решения, которые помогают оптимизировать сайт и улучшить его удобство использования. Они позволяют учитывать потребности пользователей и достигать лучших результатов в поисковых системах. Поэтому крайне важно проанализировать сайты конкурентов, на основе анализа будет создан собственный проект.

В качестве основного аналога был выбран сайт «Wildberries». Заходя на главную страницу сразу все внимание устремляется на слайдер с рекламными  предложениями, которые привлекают к себе внимание своим ярким цветом. (рисунок 1.1)

Изображение выглядит как текст, одежда, снимок экрана, человек

Автоматически созданное описание

Рисунок 1.1 – Титульная страница

Сразу на главной странице мы видим название сайта, навигационное меню, где указаны: адрес магазина, аккаунт, корзина, поиск, вакансии на работу в магазин, что является маркетинговым ходом для лучшего привлечения потенциальных клиентов.

Редакторы сайта придерживаются одного стиля во всех вкладках страницы, наполняя их максимумом информации необходимой для пользователя.

Это и стало одним из плюсов этого веб-сайта: пользователю дают всю необходимую ему информацию сразу, что делает сайт приятным и легким для пользования.

Так же на сайте присутствует раздел, где расположены контактные данные, раздел для покупателей, информация о партнёрах, проекты магазина, ссылки на социальные сети. (рисунок 1.2)

Изображение выглядит как текст, снимок экрана, фиолетовый

Автоматически созданное описание

Рисунок 1.2 – Footer

Основная цветовая палитра сайта фиолетовые цвета, такие как #7D256F, #8D297F, #BC3D96, а также #EC23BD, что позволяет выделить важные элементы на фоне других. (рисунок 1.3)

Изображение выглядит как розовый, Пурпурный цвет, Сирень, фиолетовый

Автоматически созданное описание

Рисунок 1.3 – Цветовая гамма

Разметка сайта выполнена с помощью HTML5 и CSS/SCSS, а также был использован динамический скриптовой язык программирования JavaScript. Веб-ресурс является адаптивным и кросс-браузерным, это определяется наличием различным стилей в CSS/SCSS и префиксами.

**1.2 Техническое задание**

Поставлена задача реализовать веб-сайт на тему «Сайт интернет-магазин одежды». Сайт должен быть привлекательным, содержать актуальную информацию. Данный сайт предназначен для просмотра и покупки новой одежды.

Меню навигации по сайту на всех страницах будет располагаться сверху. В меню будут предоставляться ссылки на основные разделы сайта (главная, корзина, избранное, о нас).

Почти все вышеперечисленное располагается на разных страницах для удобного перехода между ними. Раздел «Корзина» позволяет покупателям собирать и хранить выбранные ими товары перед оформлением заказа. Когда вы находите понравившуюся вам одежду или аксессуары, вы можете добавить их в корзину, а затем продолжить просмотр и выбор других товаров. Корзина отображает список выбранных товаров, и вы можете легко управлять количеством товаров, удалять или добавлять новые, а также отслеживать общую сумму заказа. После того, как вы завершили выбор товаров, вы можете перейти к оформлению заказа. Корзина в интернет-магазине одежды является удобным инструментом, который помогает вам организовать и контролировать вашу покупку перед тем, как совершить окончательный заказ. Раздел «Избранное» пользователям сохранять товары, которые им особенно нравятся или заинтересовали. Раздел "О нас" в магазине одежды представляет собой информацию о самом магазине. Это раздел, где магазин представляет себя и рассказывает о своей истории, ценностях, миссии и философии бренда.

В мобильной версии (ширина до 420 px) страница должна адаптироваться под ширину устройства. Также должна быть разработана версия для планшетного устройства (ширина до 1024px). Начиная с ширины в 1024px должна открываться версия для персонального компьютера.

Основные требования к исполнителю: создание дизайна, проведение анализа существующих решений, создание прототипа, адаптивность и кросс-браузерность сайта (корректное отображение в браузерах, таких как Google Chrome, Mozilla Firefox, Microsoft Edge, Opera.). В проекте используется гипертекстовый язык разметки HTML 5, CSS, графические элементы в формате SVG. Для хранения данных должен быть использован XML-формат.

**1.3 Выбор средств реализации программного продукта**

Первым этапом является создание макета страниц веб-сайта. Для этого использовалось приложение для разработки интерфейсов и прототипирования Figma (онлайн-сервис для дизайнеров, веб-разработчиков и маркетологов. Он предназначен для создания прототипов сайтов или приложений, иллюстраций и векторной графики. В редакторе можно настроить совместную работу, вносить и обсуждать правки, причём как в браузере, так и через приложение на компьютере). Структура сайта создана с помощью языка разметки HTML, дизайн сайта оформлен с помощью CSS. JavaScript позволит добавить динамичности сайта.

Для написания исходного кода был выбран бесплатный редактор Visual Studio Code – это бесплатное и удобное программное обеспечение, предназначенное специально для написания кода, и в частности вёрстки сайта. Для этого в программе предусмотрено множество встроенных функций, ускоряющих процесс написания кода, благодаря всплывающим подсказкам, навигации по написанному коду, автоматическому формированию отступов и встроенной функции автозаполнения. Visual Studio Code имеет отзывчивый и приятный интерфейс и возможность подключения огромного множества различных расширений (плагинов) отлично дополняющих основной функционал.

**1.4 Вывод**

В данном разделе были изучены и проанализированы различные решения, аналогичные тем, которые мы планируем использовать для создания качественного веб-сайта. Мы рассмотрели их преимущества и недостатки, чтобы определить наилучший подход для нашего проекта. Также были определены конкретные задачи, которые мы хотим достичь при создании интерактивного информационного ресурса. Мы изучили различные средства, такие как языки разметки и редактор кода Visual Studio Code, которые будут использоваться для реализации программного продукта. Мы также провели анализ особенностей языков программирования и дополнительных функций редактора кода, чтобы быть готовыми к созданию качественного продукта. Вся эта подготовка и анализ необходимы для успешного достижения наших целей и создания высококачественного продукта в дальнейшем.

**2. Проектирование страниц веб-сайта**

**2.1 Выбор способа верстки**

На сайте будет использоваться flex-box верстка, которая позволяет легко и гибко управлять расположением элементов на странице. С помощью flexbox можно создавать адаптивные и отзывчивые дизайны, а также улучшать доступность и удобство использования сайта для пользователей на различных устройствах и экранах. Flexbox позволяет легко выравнивать элементы по горизонтали и вертикали, изменять порядок расположения элементов, создавать сложные макеты и многое другое.

**2.2 Выбор стилевого оформления**

Выбор цветовой гаммы — важный этап разработки проекта. Цвета должны не только сочетаться между собой, но и отражать тему проекта. Для стилевого решения сайта были использованы белые, оранжевые, голубые и темные тона (рисунок 2.1). Белый цвет был выбран для создания чистого и простого дизайна, а голубой цвет добавлен для придания сайту свежести и жизненности. Комбинация этих двух цветов создает приятную гармонию и делает сайт более привлекательным для пользователей. Оранжевый цвет использовался для привлечения внимания, фокусировки зрения на выбранных деталях. Темные тона были подложкой для header и footer. Также на сайте используется много изображений, которые помогают визуально дополнять контент и делать его более интересным для посетителей. Красивые фотографии и иллюстрации помогают передать атмосферу и настроение сайта, а также привлекают внимание к ключевым элементам и информации. (рисунок 2.1)

**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание**

Рисунок 2.1 – Цветовая палитра

**2.3 Выбор шрифтового оформления**

В процессе создания данного макета было выбрано несколько основных шрифтов: «Koulen», «Inder», «KoHo». Все они между собой хорошо сочетаются, а также отлично подходят под выбранный стиль сайта.

**2.4 Разработка логотипа**

Логотип — это фирменный знак, который люди ассоциируют с конкретным брендом. Хороший логотип должен быть масштабируемым, разборчивым, достаточно простым, запоминающимся и узнаваемым.

Для данной веб-страницы был разработан логотип в подходящем ему стиле. (рисунок 2.2)

**Изображение выглядит как текст, Шрифт, Графика, типография

Автоматически созданное описание**

Рисунок 2.2 – Логотип сайта

**2.5 Разработка пользовательских элементов**

Пользовательскими элементами выступают навигационное меню, находящееся в шапке сайта, кроме того, необходимо уделять внимание удобству использования элементов на разных устройствах - они должны отображаться корректно и быть легко доступными как на компьютере, так и на мобильном устройстве, поэтому навигационное меню трансформируется в меню-бургер в зависимости от ширины сайта и навигационные меню, расположенные по всему сайту для более точной навигации. (рисунок 2.3)

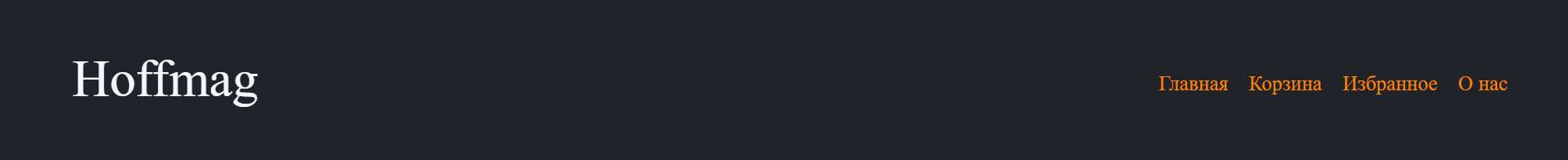
****

Рисунок 2.3 – Навигационное меню в шапке страницы

**2.6 Разработка спецэффектов**

На сайте разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Спецэффекты на странице могут значительно улучшить ее визуальный вид и сделать ее более привлекательной для пользователей. В мобильной версии сайта навигационное меню превращается в меню бургер, которое выезжает при нажатии на соответствующий значок. Иллюстрация спецэффектов представлена на рисунках 2.4-2.9.

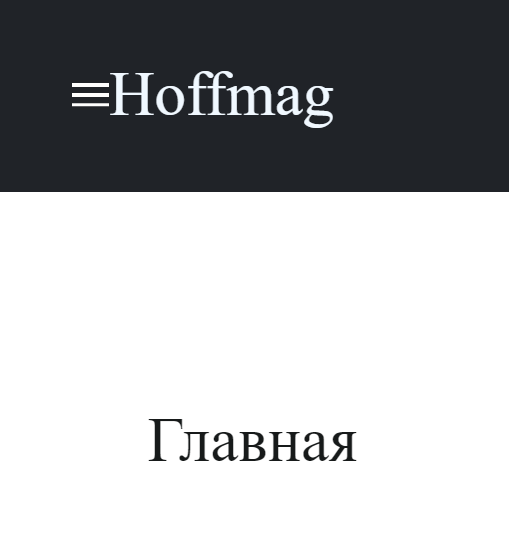


Рисунок 2.4 – Бургер меню до нажатия на кнопку

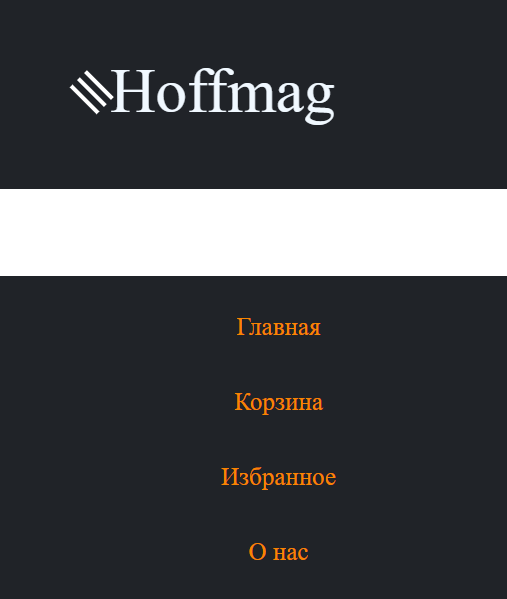


Рисунок 2.5 – Бургер меню после нажатия на кнопку

Одним из таких эффектов является динамическая анимация кнопок, связанных с добавлением товара в корзину. При наведении курсора мыши на эти кнопки, они мгновенно изменяют свой цвет, что позволяет пользователю ощутить интерактивность системы.



Рисунок 2.6 – Кнопка «В корзину» до наведения курсора мыши



Рисунок 2.7 – Кнопка «В корзину» при наведении курсора мыши



Рисунок 2.8 – Кнопка «Главная» до наведения курсора мыши

****

Рисунок 2.9 – Кнопка «Главная» при наведении курсора мыши

Разработка спецэффектов для сайта — это мощный инструмент веб-дизайна, который может значительно улучшить визуальную привлекательность, интерактивность и функциональность веб-ресурса. Правильно реализованные спецэффекты помогают создать уникальный и запоминающийся пользовательский опыт, повышая удовлетворенность пользователей и конверсии.

**2.7. Вывод**

В ходе выполнения работы было изучено множество аспектов проектирования веб-сайтов, начиная от выбора способа верстки и заканчивая разработкой спецэффектов. Были рассмотрены современные тенденции и тренды в дизайне, а также принципы удобства использования и доступности сайта. В результате был разработан дизайн сайта, учитывающий все вышеупомянутые аспекты. В целом, выполнение работы позволило получить ценный опыт в проектировании веб-сайтов и подготовиться к созданию продукта на практике.

**3. Реализация структуры веб-сайта**

**3.1 Структура HTML-документа**

В курсовой работе необходимо разработать структуру HTML-документа, должна отражать все элементы сайта и определять их расположение на странице.

При разработке структуры HTML-документа необходимо учитывать принципы семантической верстки. Это означает, что каждый элемент должен быть определен с использованием соответствующего тега, который отражает его смысл и функциональность.

Структура HTML-документа состоит из нескольких основных элементов:

1. Тег <html> – определяет начало и конец HTML-документа.

2. Тег <head> – внутри этого тега размещаются метаданные документа, такие как заголовок страницы, ключевые слова для поисковых систем, описание страницы, подключение внешних файлов (например, стилей или скриптов) и другая информация, которая не отображается на самой странице. Пример тега <head> представлен на листинге 3.1.

|  |
| --- |
| <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Hoffmag</title>  <link rel = "stylesheet" href="header-footer.css">  <link rel = "stylesheet" href = "info-style.css">  </head> |

Листинг 3.1 – Тег <head>

3. Тег <body> – определяет основное содержимое страницы, включая текст, изображения, видео и другие элементы.

4. Тег <header> – внутри этого тега размещаются заголовок страницы, логотип и другие элементы, которые должны быть видны на каждой странице сайта. Например, это может быть верхняя панель навигации или общая информация о сайте. Пример тега < header > представлен на листинге 3.2.

|  |
| --- |
| <header>  <div class = "title">  <a href = "index.html">Hoffmag</a>  </div>  <div class="menu-toggle">  <div class="hamburger"></div>  </div>  <ul class = "nav">  <li>  <a href = "index.html">Главная</a>  </li>  <li>  <a href = "korzina.html">Корзина</a>  </li>  <li>  <a href = "Loved.html">Избранное</a>  </li>  <li>  <a href = "index.html#footer">О нас</a>  </li>  </ul>  </header> |

Листинг 3.2 – Тег <header>

5. Тег <nav> – определяет навигационное меню сайта.

6. Тег <main> – определяет главное содержимое страницы.

7. Тег <section> – определяет разделы страницы.

8. Тег <article> – определяет отдельные статьи или записи на странице.

9. Тег <aside> – определяет боковую панель с дополнительной информацией, такой как реклама или ссылки на другие страницы сайта.

10. Тег <footer> – содержит информацию об авторах, дате создания страницы, ссылки на социальные сети и другую дополнительную информацию. Пример тега < footer > представлен на листинге 3.3.

|  |
| --- |
| <footer id="footer">  <div class="footer-content">  <div class="footer-section">  <h2>Покупателям</h2>  <ul>  <li><a href = "faq.html">Часто задаваемые вопросы</a></li>  </ul>  </div>  <div class="footer-section">  <h2>Мы в соцсетях</h2>  <ul>  <li><a href = "https://vk.com/feed">Вконтакте</a></li>  <li><a href = "https://www.instagram.com/it.belstu/">Инстаграм</a></li>  <li><a href = "https://telegram.org/">Телеграм</a></li>  </ul>  </div>  <div class="footer-section">  <h2>Меню</h2>  <ul>  <li><a href = "index.html">Главная</a></li>  <li><a href = "Korzina.html">Корзина</a></li>  </ul>  </div>  <div class="footer-section" data-section="4">  </div>  </div>  <script src = "addsize.js"></script>  <script src = "xml.js"></script>  </footer> |

Листинг 3.3 – Содержание тега <footer> главной страницы

В начале документа в теге <head></head> прописана служебная информация, название веб-сайта. Тег <head> предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера <head> находятся метатеги, которые используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.

Содержимое тега <head> не отображается напрямую на веб-странице, за исключением тега <title> устанавливающего заголовок окна веб-страницы.

Далее расположен тег <body></body>, внутри которого размещаются все теги, в которых расположен сайт.

В теге header содержится контактная информация, логотип сайта и горизонтальное навигационное меню.

В теге footer содержится контактная информация, логотип сайта и меню для связи.

**3.2 Добавление таблиц стилей CSS**

Добавление таблиц стилей CSS позволяет упростить создание стилистического оформления страницы.

В данной структуре используется внешнее подключение таблиц стилей через тег link, так как это самый мощный и удобный способ определения стилей и правил для веб-сайта. Стили хранятся в отдельном файле, который может быть использован для любых веб-страниц. В таблицах стилей используются селекторы по классу, по идентификатору, а также селекторы потомков, псевдоклассы и псевдоэлементы.

Для разных частей шапки сайта  использован следующий код CSS. В листингах 3.4—3.6 представлены примеры использования CSS.

|  |
| --- |
| @media only screen and (max-width: 767px) {      .catalog {          display: flex;          flex-direction: column; /\* Переключаем на вертикальное расположение элементов \*/      }      .item {          margin-bottom: 20px; /\* Добавляем отступ снизу между элементами \*/      }      .item img {          width: 100%; /\* Заполняем доступное пространство \*/      }      .text {          text-align: center; /\* Выравниваем текст по центру \*/      }      .cost {          font-size: 16px; /\* Уменьшаем размер шрифта для цены \*/      }      .sizes {          margin-bottom: 10px; /\* Добавляем отступ снизу между размерами и типом товара \*/      }      .bean-button {          margin-top: 10px; /\* Добавляем отступ сверху для кнопки \*/      }  } |

Листинг 3.4 – CSS код медиа запроса

Использование CSS для header представлено на листинге 3.5.

|  |
| --- |
| header{  position: relative;  width: 100%;  height: 154px;  left: 0px;  top: 0px;  background-color: #202328;  z-index: 10;  } |

Листинг 3.5 – CSS код для <header>

Использование CSS для footer представлено на листинге 3.6.

|  |
| --- |
| footer {      background-color: #202328;      color: #FFFFFF;      padding: 40px 0;      width: 100%;      height: auto;      z-index: 10;      bottom: 0;  }  .footer-content {      display: flex;      flex-direction: column; /\* Change to column layout on smaller screens \*/      justify-content: space-between;      padding: 0 20px; /\* Adjust padding for smaller screens \*/  }  .footer-section {      width: 100%; /\* Occupy full width on smaller screens \*/      margin-bottom: 20px; /\* Add some spacing between sections \*/  }  .footer-section h2 {      font-family: 'Koulen';      font-style: normal;      font-weight: 400;      font-size: 35px;      line-height: 1;      margin-bottom: 10px;      color: #F88500;  }  .footer-section ul {      list-style-type: none;      padding: 0;  }  .footer-section li {      margin-bottom: 10px;  }  .footer-section a {      text-decoration: none;      color: aliceblue;  } |

Листинг 3.6 – CSS код для <footer>

В заключение, CSS используется для определения внешнего вида и стиля веб-страницы. Он позволяет разработчикам контролировать различные аспекты дизайна, такие как цвета, шрифты, размеры, расположение элементов и многое другое.

**3.3 Использование стандартов XML (SVG)**

Использование стандартов XML (Extensible Markup Language) и SVG (Scalable Vector Graphics) в веб-разработке предоставляет возможность создания и отображения векторной графики на веб-страницах.

XML является универсальным языком разметки, используемым для описания структурированных данных. Он позволяет создавать пользовательские теги и определять свою собственную семантику для данных. XML широко применяется в обмене данных между различными системами и веб-сервисами.

Для хранения конткактной информации в footer использовался XML. Данные конткактной информации удобно хранить отдельно в XML-документах.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <root>      <footer-section id="4">          <title>О нас</title>          <description>Наш интернет-магазин Hoffmag рад привествовать вас на данной платформе. Надеемся вы подберет что-нибудь себе по вкусу </description>          <support>Круглосуточная поддержка</support>          <phone>+375 (33) 123-123-23</phone>      </footer-section>  </root> |

Листинг 3.7 – Использование XML

SVG является форматом для описания векторной графики с использованием XML-разметки. Он позволяет создавать масштабируемые и интерактивные изображения, которые могут быть отображены на веб-страницах без потери качества при изменении размеров. SVG поддерживает различные элементы графики, такие как линии, кривые, примитивы, текст, градиенты и трансформации. На листинге 3.8 представлен пример использования SVG.

|  |
| --- |
| <svg width="41" height="33" viewBox="0 0 41 33" fill="none" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">  <rect width="41" height="33" fill="url(#pattern0\_63\_344)" onclick="addToLiked(event)"/>  <defs>  <pattern id="pattern0\_63\_344" patternContentUnits="objectBoundingBox" width="1" height="1">  <use xlink:href="#image0\_63\_344" transform="matrix(0.00277778 0 0 0.00345118 0 -0.121212)"/>  </pattern>  <image id="image0\_63\_344" width="360" height="360" xlink:href="data:image/png;base64"/>  </defs>  </svg> |

Листинг 3.8 – Использование SVG

SVG — это мощный и гибкий формат для векторной графики, который широко используется в веб-дизайне благодаря своей масштабируемости, малому размеру файлов, редактируемости и поддержке анимации. Эти характеристики делают SVG незаменимым инструментом для создания высококачественной графики, которая отлично выглядит на любом устройстве и экране.

**3.4 Управление элементами DOM**

Управление элементами DOM (Document Object Model) в веб-разработке позволяет изменять содержимое, структуру и стили веб-страницы с использованием JavaScript. DOM представляет собой представление веб-страницы в виде древовидной структуры, где каждый элемент HTML представлен в виде объекта, доступного для манипуляции.

JavaScript – это язык программирования, который используют разработчики для создания интерактивных веб-страниц. Функции JavaScript могут улучшить удобство взаимодействия пользователя с веб-сайтом: от обновления ленты новостей в социальных сетях и до отображения анимации и интерактивных карт.

Может подключаться как внутри самого HTML-документа, так и отдельным файлом.

В данном проекте JavaScript был использован для меню, слайдера для прокрутки фотографий, кнопок, переводящих на другие разделы сайта и тд. На листингах 3.9 – 3.10 представлены примеры использования JS.

|  |
| --- |
| window.addEventListener("load", () => {      let temp = [];      let name, price, picture, item, size;      for (let i = 1; i < sessionStorage.length; i = i + 4) {        if (i == 1) {          name = sessionStorage.getItem(`${i}name`);          price = sessionStorage.getItem(`${i}price`);          picture = sessionStorage.getItem(`${i}picture`);          size = sessionStorage.getItem(`${i}size`);        } else {          name = sessionStorage.getItem(`${parseInt(i + 3) / 4}name`);          price = sessionStorage.getItem(`${parseInt(i + 3) / 4}price`);          picture = sessionStorage.getItem(`${parseInt(i + 3) / 4}picture`);          size = sessionStorage.getItem(`${parseInt(i + 3) / 4}size`);        }        item = { name: name, price: price, picture: picture, size: size };        temp.push(item);        console.log(temp)      }        const main = document.querySelector(".items-container");        temp.forEach((item) => {        const node = document.createElement("div");        node.classList.add("item");        //   const imageContainer = document.createElement("div");      //   imageContainer.classList.add("game-image");        const image = document.createElement("img");        image.src = item.picture;      //   imageContainer.appendChild(image);        node.appendChild(image);        let info =document.createElement("div")        info.classList.add("item-info")        let price=document.createElement("p");        price.innerHTML=`Цена: ${item.price}`;        let name=document.createElement("p");        name.innerHTML=item.name;        let size=document.createElement("p");        size.innerHTML=`Размер: ${item.size}`;        info.appendChild(price)        info.appendChild(name)        info.appendChild(size)        node.appendChild(info)        let qc=document.createElement("div")        qc.classList.add("quantity-counter")        let qd=document.createElement("button")        qd.classList.add("quantity-decrease")        qd.innerHTML="-"        let q=document.createElement("div")        q.classList.add("quantity")        q.innerHTML="1"        let qi=document.createElement("button")        qi.classList.add("quantity-inccrease")        qi.innerHTML="+"        qc.appendChild(qd)        qc.appendChild(q)        qc.appendChild(qi)        info.appendChild(qc)        let deleteButton=document.createElement("button")        deleteButton.innerHTML="X"        deleteButton.classList.add("deleet-btn")        node.appendChild(deleteButton)          deleteButton.addEventListener("click", deleteFromBean)          main.appendChild(node);      });    });    //   let delButtons=Array.from(document.getElementsByClassName("del"))  //   delButtons.forEach((button)=>{button.addEventListener('click', deleteFromBean)})        function deleteFromBean(event) {        console.log("s")        let arr = document.getElementsByClassName("del");        let index;        for (let i = 0; i < arr.length; i++) {          if (arr[i] === event.target) {            index = i;          }        }          let name = document.getElementsByClassName("game-title")[index].innerHTML;          let gamesContainersArray = Array.from(document.getElementsByClassName("game-container"));        let gamesArray = Array.from(document.getElementsByClassName("game-title"));        console.log(gamesArray)        let deleteIndex;        for (let i = 0; i < gamesArray.length; i++) {          if (gamesArray[i].innerHTML === name) {            deleteIndex = i;            console.log(gamesArray[i].innerHTML)          }        }        console.log(name)        console.log(deleteIndex)        console.log(gamesContainersArray)        gamesContainersArray[deleteIndex].parentNode.removeChild(gamesContainersArray[deleteIndex])        let str;        for (let i = 0; i < sessionStorage.length; i++) {            str = i;            if (sessionStorage[`${str}name`] === name) {                break;            }          }          console.log(str)      for(; str<sessionStorage.length/3; str++){        sessionStorage.setItem(`${str}name`, sessionStorage[`${str+1}name`] )      }    sessionStorage.removeItem(`${str}name`)    sessionStorage.removeItem(`${str}price`)    sessionStorage.removeItem(`${str}picture`)    } |

Листинг 3.9 – Пример использования JS кода для работы корзины

Использование JS для выбора размера представлено на листинге 3.10.

|  |
| --- |
| var sizes = document.querySelectorAll('.size');  // Перебираем каждый элемент  sizes.forEach(function(size) {      // Добавляем обработчик события клика      size.addEventListener('click', function() {          // Убираем класс active у всех размеров          sizes.forEach(function(size) {              size.classList.remove('active');          });          // Добавляем класс active только к текущему выбранному размеру          size.classList.add('active');      });  }); |

Листинг 3.10 – Пример использования JS кода для выбора размера

JavaScript — это универсальный язык программирования, который является неотъемлемой частью современного веба. Его способность добавлять интерактивность, мощная экосистема библиотек и фреймворков, а также широкая поддержка делают его незаменимым инструментом для разработчиков. Благодаря JavaScript веб-страницы и приложения становятся более динамичными, отзывчивыми и привлекательными для пользователей.

**3.5. Вывод**

На данном этапе была реализована структура на HTML, созданы таблицы стилей CSS. Было продемонстрировано использование стандартов XML (SVG) и языка программирования JS. Определено стилевое оформление веб-сайта, созданы анимации, а также веб-сайт был подготовлен к следующему этапу – тестированию.

**4. Тестирование веб-сайта**

**4.1 Адаптивный дизайн веб-сайта**

Адаптивная вёрстка сайта — это способ обеспечить оптимальное отображение интерфейсов на устройствах с разным разрешением экранов.

Адаптивный дизайн предусматривает, что пользователь будет взаимодействовать с кнопками, текстом, изображениями через разные устройства. Думает об удобстве пользователя в этом случае уже дизайнер.

Для адаптива данного проекта была выбрана flex вёрстка и медиа-запросы. Адаптив должен был быть сделан для компьютерной версии (1920px – 1200px), планшета (1199px-960px), телефона (959px – 320px).

Пример адаптивной верстки представлен в приложении на рисунках 4.1 – 4.3.

Изображение выглядит как одежда, снимок экрана, текст, женщина

Автоматически созданное описание

Рисунок 4.1 - Адаптивность для компьютерной версии



Рисунок 4.2 - Адаптивность для планшета

Изображение выглядит как текст, Человеческое лицо, снимок экрана, одежда

Автоматически созданное описание

Рисунок 4.3 - Адаптивность для телефона

Адаптивность сайта — это необходимый стандарт в современном веб-дизайне, который обеспечивает удобство использования и доступность веб-ресурсов на всех устройствах. Благодаря адаптивности сайты становятся более привлекательными и удобными для пользователей, что повышает уровень их удовлетворенности и удержания. Кроме того, адаптивный дизайн улучшает позиции сайта в поисковых системах, что способствует увеличению трафика и конверсий. Внедрение адаптивного дизайна — это инвестиция в будущее вашего веб-ресурса, позволяющая создать устойчивый и конкурентоспособный сайт.

**4.2 Кроссбраузерность веб-сайта**

Кроссбраузерность — это способность веб-страницы работать одинаково хорошо на разных браузерах и платформах. Использование SVG на нашем сайте также помогает обеспечить кроссбраузерность, поскольку формат поддерживается большинством современных браузеров, включая Chrome, Firefox и Edge.

Веб-сайт был открыт при помощи различных браузеров:

1. Microsoft Edge (Рисунок 4.4)
2. Google Chrome (Рисунок 4.5)
3. Mozilla Firefox (Рисунок 4.6)

Изображение выглядит как одежда, человек, женщина, платье

Автоматически созданное описание

Рисунок 4.4 - Главная страница в Microsoft Edge

Изображение выглядит как одежда, человек, снимок экрана, женщина

Автоматически созданное описание

Рисунок 4.5 - Главная страница в Google Chrome

Изображение выглядит как одежда, человек, женщина, снимок экрана

Автоматически созданное описание

Рисунок 4.6 - Главная страница в Mozilla Firefox

Кроссбраузерность сайта — это фундаментальный аспект качественной веб-разработки, обеспечивающий доступность и удобство использования веб-ресурсов для широкой аудитории. Поддержание кроссбраузерной совместимости помогает улучшить пользовательский опыт, расширить аудиторию и укрепить доверие к сайту. Инвестирование в кроссбраузерное тестирование и следование стандартам веб-разработки позволяют создавать устойчивые и конкурентоспособные веб-решения, которые остаются актуальными и функциональными в условиях быстро меняющегося технологического ландшафта.

**4.3 Руководство пользователя**

Сайт магазина одежды "Hoffmag" предоставляет пользователям актуальную информацию о товарах, коллекциях, акциях и услугах, предлагаемых магазином. Он также предоставляет возможность клиентам просматривать фотографии товаров для привлечения их внимания. Главная функция сайта - предоставить удобный и информативный интерфейс, позволяющий пользователям легко найти нужные разделы и товары.

Сайт включает в себя разделы: главная страница, корзина, избранное, карточки товара и контакты. Каждая страница имеет горизонтальное меню, которое позволяет пользователям быстро переходить между разделами. Для создания более привлекательного интерфейса, пункты меню анимированы в стиле сайта, что делает использование сайта более приятным для пользователей.

**4.4 Тестирование кода**

В процессе разработки веб-сайта, тестирование кода играет критическую роль. Оно становится важным этапом, который позволяет тщательно проверить каждый аспект веб-проекта, начиная от его визуального представления и заканчивая функциональными возможностями.

Первое, что необходимо учитывать — это корректность отображения веб-страниц на различных устройствах и в разных браузерах. Каждый браузер может по-разному интерпретировать код, что может привести к непредвиденным отображениям и расхождениям в дизайне. Проведение тестирования поможет обнаружить такие проблемы и гарантировать, что сайт выглядит одинаково привлекательно и функционально на всех платформах.

Для тестирования кода был выбран сервис The W3C Markup Validation Service. Результат тестирования представлен на рисунке 4.7.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 4.7 – Результат тестирования кода

**4.5 Вывод**

На основе проведенного тестирования, я рад сообщить, что веб-сайт магазина одежды "Hoffmag" обладает эффективным адаптивным дизайном. Это означает, что пользователи могут комфортно и удобно просматривать сайт на различных устройствах, включая компьютеры, планшеты и мобильные телефоны. Это важно, так как люди используют разные устройства для доступа к интернету, и сайт должен корректно отображаться на всех из них.

Кроме того, сайт магазина одежды "Hoffmag" обеспечивает кроссбраузерность, что означает, что он может работать в различных веб-браузерах, таких как Chrome, Firefox, Safari и других. Это позволяет пользователям выбирать браузер, который им наиболее удобен, и все равно получать полноценный доступ к сайту.

Руководство пользователя, предоставляет необходимую информацию и инструкции, чтобы помочь пользователям максимально эффективно использовать сайт. Это включает в себя информацию о различных разделах, услугах, акциях и инструкции о том, как сделать покупку или связаться с магазином. Это помогает пользователям легко ориентироваться на сайте и получать необходимую помощь.

Все эти факторы способствуют удобству использования сайта, улучшению пользовательского опыта и удовлетворенности клиентов магазина одежды "Hoffmag".

**Заключение**

В заключение, сайт для магазина одежды "Hoffmag" является важным и полезным ресурсом, предоставляющим пользователям свежую и актуальную информацию о товарах, коллекциях, акциях и услугах магазина. Он обеспечивает все необходимые функциональные возможности, которые могут потребоваться пользователям при выборе и покупке одежды.

Для создания качественного сайта необходимо уделить внимание не только контенту, но и дизайну, функциональности и удобству использования для пользователей. Важно также обеспечить адаптивность сайта и оптимизацию под различные устройства, чтобы пользователи могли просматривать и покупать товары на любом устройстве.

Для реализации макета веб-страницы были исследованы другие сайты в сфере моды и магазинов одежды, чтобы учесть современные тенденции и лучшие практики.

Для удобного написания кода веб-страницы была использована среда разработки VS Code, с установленными плагинами для повышения эффективности работы.

Перед написанием кода были изучены новые стандарты языков разметки и программирования, а также прочитаны документации по использованию технологий, необходимых для создания сайта магазина одежды.

После написания всего сайта проводилось тестирование, в ходе которого выявлялись и устранялись недочеты, чтобы обеспечить высокое качество и функциональность сайта.

Реализация важных элементов страницы или скриптов представлялась в виде приложений, которые давали общую информацию о продукте и помогали пользователям сделать правильный выбор.

В целом, для выполнения всех поставленных задач на этом проекте были использованы как теоретические знания, так и практические навыки разработки веб-сайтов, чтобы создать удобный и привлекательный ресурс для наших клиентов.

**Использованные литературные источники**

1. Аналог «Zalando» [Электронный ресурс] – Режим доступа: https://www.zalando.co.uk/

2. Приложение для разработки макетов и прототипов веб-страниц «Figma» [Электронный ресурс]. – Режим доступа: https://www.figma.com/

3. Аналог «Bohoo» [Электронный ресурс] – Режим доступа: https://www.boohoo.com/

4. Справочник по HTML [Электронный ресурс] – Режим доступа: https://htmlbook.ru/html

5. Документация по SCSS [Электронный ресурс] – Режим доступа: https://sass-scss.ru/

6. Документация по HTML/CSS/JavaScript [Электронный ресурс] – Режим доступа: https://developer.mozilla.org/

7. Репозиторий проекта на GitHub [Электронный ресурс] — Режим доступа: https://github.com/varvvk/-The-Shoes-web-site.git

8. Киселенко Н. П. HTML. Самое необходимое – Санкт-Петербург: БХВ-Петербург, 2008. – 352 с.

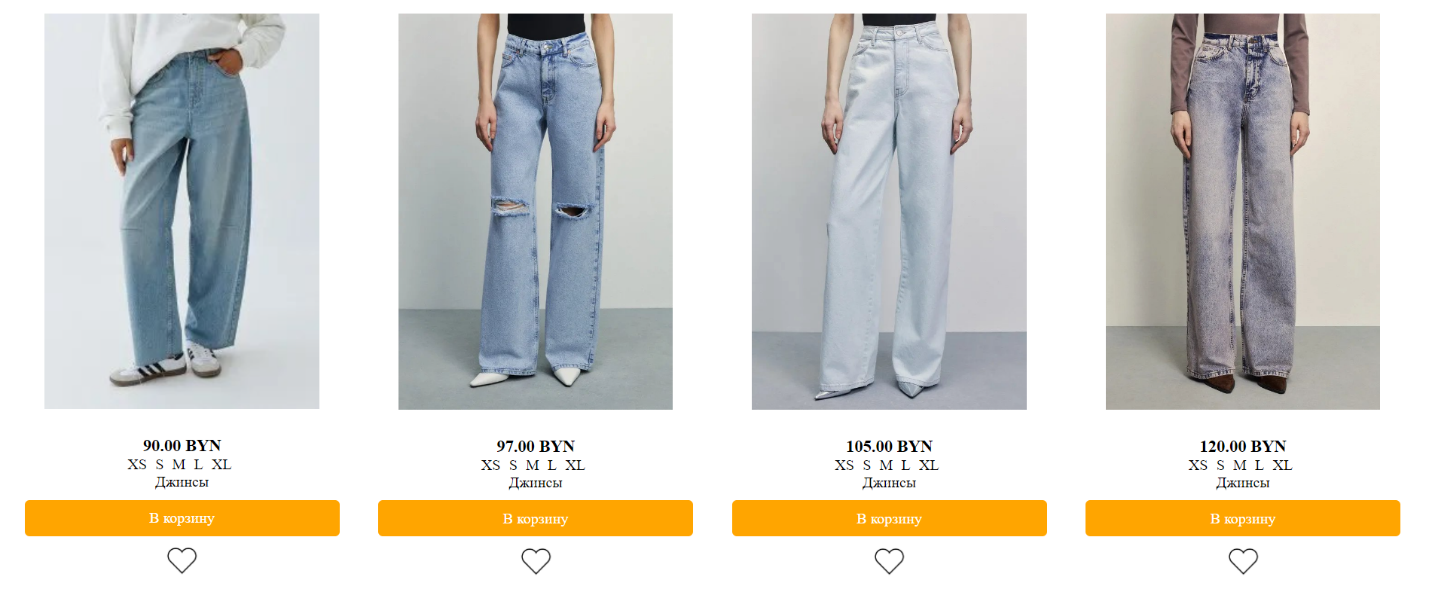
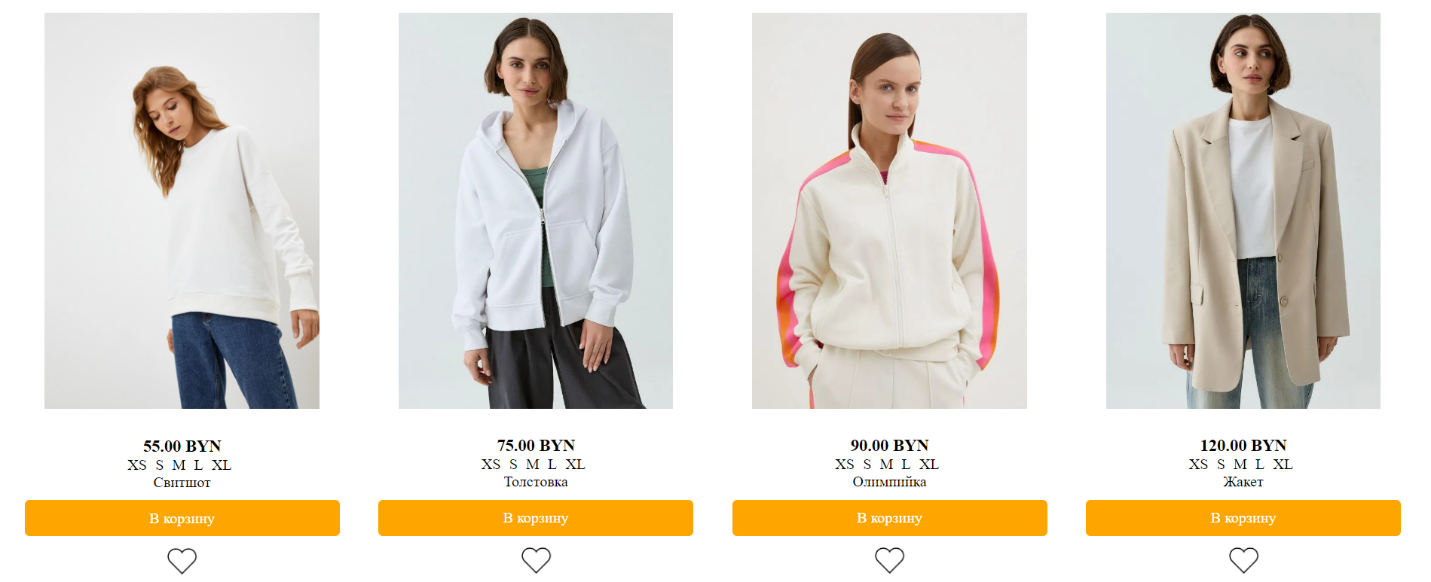
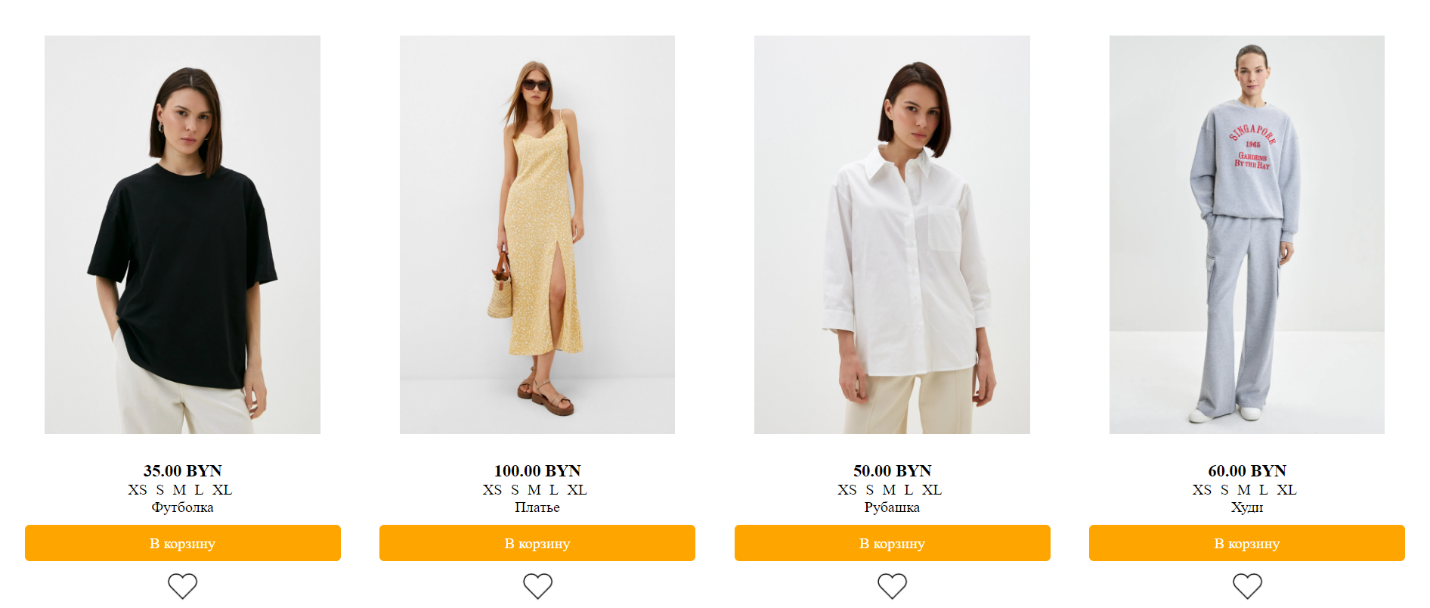
9. Репозиторий проекта на GitHub [Электронный ресурс] — Режим доступа:

**Приложение А**

Макет страниц

Главная страница

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описаниеИзображение выглядит как одежда, женщина, Дизайн одежды, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Страница товара

Изображение выглядит как текст, снимок экрана, одежда, человек

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Страница корзины

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Страница избранного

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Страница часто задаваемых вопросов

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Приложение Б**

Листинг HTML-документа

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Hoffmag</title>      <link rel = "stylesheet" href="header-footer.css">      <link rel = "stylesheet" href = "main-style.css">  </head>  <body>      <script src = "hamburger.js"></script>      <header>          <div class = "title">              <a href = "index.html">Hoffmag</a>              </div>              <div class="menu-toggle">                  <div class="hamburger"></div>          </div>          <ul class = "nav">              <li>                  <a href = "index.html">Главная</a>              </li>              <li>                  <a href = "korzina.html">Корзина</a>              </li>              <li>                  <a href = "Loved.html">Избранное</a>              </li>              <li>                  <a href = "index.html#footer">О нас</a>              </li>          </ul>      </header>  <footer id="footer">              <div class="footer-content">                  <div class="footer-section">                      <h2>Покупателям</h2>                      <ul>                          <li><a href = "faq.html">Часто задаваемые вопросы</a></li>                      </ul>                  </div>                  <div class="footer-section">                      <h2>Мы в соцсетях</h2>                      <ul>                          <li><a href = "https://vk.com/feed">Вконтакте</a></li>                          <li><a href = "https://www.instagram.com/it.belstu/">Инстаграм</a></li>                          <li><a href = "https://telegram.org/">Телеграм</a></li>                      </ul>                  </div>                  <div class="footer-section">                      <h2>Меню</h2>                      <ul>                          <li><a href = "index.html">Главная</a></li>                          <li><a href = "Korzina.html">Корзина</a></li>                      </ul>                  </div>                  <div class="footer-section" data-section="4">                  </div>              </div>              <script src = "addsize.js"></script>              <script src = "xml.js"></script>      </footer>      <script src="index.js"></script>  </body>  </html> |

**Приложение В**

Листинг CSS

|  |
| --- |
| body {      max-height: 100%;      margin: 0; /\* Убираем отступы \*/      padding: 0; /\* Убираем внутренние отступы \*/      overflow-x: hidden;  }  main {      margin-top:154px;      margin-bottom: 350px;  }  header{  position: relative;  width: 100%;  height: 154px;  left: 0px;  top: 0px;  background-color: #202328;  z-index: 10;  }  .title{  position: absolute;  width: 201px;  height: 63px;  left: 70px;  top: 45px;  font-family: 'Inder';  font-style: normal;  font-weight: 400;  font-size: 50px;  line-height: 62px;  display: flex;  align-items: center;  text-align: center;  color: #FFFFFF;  }  a:hover{      text-shadow: 0 0 50px #fff,                    0 0 12px #fff;  }  a{      text-decoration: none;      color: aliceblue;      transition: all 0.8s ease;  }  /\* Frame 2 \*/  /\* Auto layout \*/  .nav {      position: absolute;      margin-top: 70px; /\* Adjust as needed \*/      right: 60px; /\* Adjust as needed \*/      display: flex;      flex-direction: row;      justify-content: flex-end;      align-items: center;      padding: 0;      gap: 20px; /\* Increase the gap between buttons \*/      list-style-type: none;  }  /\* Главная \*/  .nav li {      margin: 0;      width: auto;      height: auto;      font-family: 'Koulen';      font-style: normal;      font-weight: 400;      font-size: 20px;      line-height: 1;  }  li a {      color:#F88500;  }  /\* Footer\*/  /\* Footer styles \*/  footer {      background-color: #202328;      color: #FFFFFF;      padding: 40px 0;      width: 100%;      height: auto;      z-index: 10;      bottom: 0;  }  .footer-content {      display: flex;      flex-direction: column; /\* Change to column layout on smaller screens \*/      justify-content: space-between;      padding: 0 20px; /\* Adjust padding for smaller screens \*/  }  .footer-section {      width: 100%; /\* Occupy full width on smaller screens \*/      margin-bottom: 20px; /\* Add some spacing between sections \*/  }  .footer-section h2 {      font-family: 'Koulen';      font-style: normal;      font-weight: 400;      font-size: 35px;      line-height: 1;      margin-bottom: 10px;      color: #F88500;  }  .footer-section ul {      list-style-type: none;      padding: 0;  }  .footer-section li {      margin-bottom: 10px;  }  .footer-section a {      text-decoration: none;      color: aliceblue;  }  /\* Media query for responsive design \*/  @media only screen and (min-width: 768px) {          .footer-content {          flex-direction: row;      }        .footer-section {          width: 25%;          margin-bottom: 0;      }  }  /\* Add hamburger menu styles \*/  .menu-toggle {      display: none;      position: absolute; /\* Позиционируем абсолютно, чтобы иметь контроль над положением \*/      top: 50%; /\* Размещаем на 50% от верхней грани \*/      left: 20px; /\* Отступ справа \*/      transform: translateY(-50%); /\* Сдвигаем на половину высоты элемента вверх, чтобы центрировать вертикально \*/  }  .hamburger {      width: 30px;      height: 3px;      background-color: #FFFFFF;      margin: 6px 0;      transition: transform 0.3s ease;  }  .hamburger.active {      transform: rotate(45deg);  }  .hamburger::before,  .hamburger::after {      content: '';      display: block;      width: 30px;      height: 3px;      background-color: #FFFFFF;      transition: transform 0.3s ease;  }  .hamburger::before {      transform: translateY(-8px);  }  .hamburger::after {      transform: translateY(5px);  }  /\* Media query for responsive design \*/  @media only screen and (max-width: 768px) {      /\* For tablets and smaller screens \*/      .title{          display: none;      }      .menu-toggle {          display: block; /\* Display the menu toggle on smaller screens \*/          cursor: pointer;      }      .nav {          display: none; /\* Hide the navigation links by default on smaller screens \*/      }      .nav.active {          display: flex; /\* Display the navigation links when active (clicked) \*/          flex-direction: column;          background-color: #202328;          position: absolute;          top: 154px; /\* Adjust as needed \*/          left: 0;          width: 100%;          padding: 20px;          z-index: 9;      }      .nav li {          margin: 10px 0;      }  }  main h2 {      position: relative;      font-family: 'Koulen';      font-style: normal;      font-weight: 400;      font-size: 50px;      line-height: 90px;      display: flex;      justify-content: center;      align-items: center;      color: #171A1A;      mix-blend-mode: plus-darker;  } |

**Приложение Г**

Листинг SVG

|  |
| --- |
| <svg width="41" height="33" viewBox="0 0 41 33" fill="none" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">                          <rect width="41" height="33" fill="url(#pattern0\_63\_344)" onclick="addToLiked(event)"/>                          <defs>                          <pattern id="pattern0\_63\_344" patternContentUnits="objectBoundingBox" width="1" height="1">                          <use xlink:href="#image0\_63\_344" transform="matrix(0.00277778 0 0 0.00345118 0 -0.121212)"/>                          </pattern>                          <image id="image0\_63\_344" width="360" height="360" xlink:href="data:image/png;base64"/>                          </defs>                          </svg> |

**Приложение Д**

Листинг JavaScrpit

|  |
| --- |
| window.addEventListener("load", () => {      let temp = [];      let name, price, picture, item, size;      for (let i = 1; i < sessionStorage.length; i = i + 4) {        if (i == 1) {          name = sessionStorage.getItem(`${i}name`);          price = sessionStorage.getItem(`${i}price`);          picture = sessionStorage.getItem(`${i}picture`);          size = sessionStorage.getItem(`${i}size`);        } else {          name = sessionStorage.getItem(`${parseInt(i + 3) / 4}name`);          price = sessionStorage.getItem(`${parseInt(i + 3) / 4}price`);          picture = sessionStorage.getItem(`${parseInt(i + 3) / 4}picture`);          size = sessionStorage.getItem(`${parseInt(i + 3) / 4}size`);        }        item = { name: name, price: price, picture: picture, size: size };        temp.push(item);        console.log(temp)      }        const main = document.querySelector(".items-container");        temp.forEach((item) => {        const node = document.createElement("div");        node.classList.add("item");        //   const imageContainer = document.createElement("div");      //   imageContainer.classList.add("game-image");        const image = document.createElement("img");        image.src = item.picture;      //   imageContainer.appendChild(image);        node.appendChild(image);        let info =document.createElement("div")        info.classList.add("item-info")        let price=document.createElement("p");        price.innerHTML=`Цена: ${item.price}`;        let name=document.createElement("p");        name.innerHTML=item.name;        let size=document.createElement("p");        size.innerHTML=`Размер: ${item.size}`;        info.appendChild(price)        info.appendChild(name)        info.appendChild(size)        node.appendChild(info)        let qc=document.createElement("div")        qc.classList.add("quantity-counter")        let qd=document.createElement("button")        qd.classList.add("quantity-decrease")        qd.innerHTML="-"        let q=document.createElement("div")        q.classList.add("quantity")        q.innerHTML="1"        let qi=document.createElement("button")        qi.classList.add("quantity-inccrease")        qi.innerHTML="+"        qc.appendChild(qd)        qc.appendChild(q)        qc.appendChild(qi)        info.appendChild(qc)        let deleteButton=document.createElement("button")        deleteButton.innerHTML="X"        deleteButton.classList.add("deleet-btn")        node.appendChild(deleteButton)          deleteButton.addEventListener("click", deleteFromBean)          main.appendChild(node);      });    });    //   let delButtons=Array.from(document.getElementsByClassName("del"))  //   delButtons.forEach((button)=>{button.addEventListener('click', deleteFromBean)})        function deleteFromBean(event) {        console.log("s")        let arr = document.getElementsByClassName("del");        let index;        for (let i = 0; i < arr.length; i++) {          if (arr[i] === event.target) {            index = i;          }        }          let name = document.getElementsByClassName("game-title")[index].innerHTML;          let gamesContainersArray = Array.from(document.getElementsByClassName("game-container"));        let gamesArray = Array.from(document.getElementsByClassName("game-title"));        console.log(gamesArray)        let deleteIndex;        for (let i = 0; i < gamesArray.length; i++) {          if (gamesArray[i].innerHTML === name) {            deleteIndex = i;            console.log(gamesArray[i].innerHTML)          }        }        console.log(name)        console.log(deleteIndex)        console.log(gamesContainersArray)        gamesContainersArray[deleteIndex].parentNode.removeChild(gamesContainersArray[deleteIndex])        let str;        for (let i = 0; i < sessionStorage.length; i++) {            str = i;            if (sessionStorage[`${str}name`] === name) {                break;            }          }          console.log(str)      for(; str<sessionStorage.length/3; str++){        sessionStorage.setItem(`${str}name`, sessionStorage[`${str+1}name`] )      }    sessionStorage.removeItem(`${str}name`)    sessionStorage.removeItem(`${str}price`)    sessionStorage.removeItem(`${str}picture`)    } |

|  |
| --- |
| // Получаем все элементы с классом .size  var sizes = document.querySelectorAll('.size');  // Перебираем каждый элемент  sizes.forEach(function(size) {      // Добавляем обработчик события клика      size.addEventListener('click', function() {          // Убираем класс active у всех размеров          sizes.forEach(function(size) {              size.classList.remove('active');          });          // Добавляем класс active только к текущему выбранному размеру          size.classList.add('active');      });  }); |

|  |
| --- |
| // function deleteFromBean(str, event) {  //     console.log("F")    // for(; str<sessionStorage.length/4; str++){  //     sessionStorage.setItem(`${str}name`, sessionStorage[`${str+1}name`] )  // }  // sessionStorage.removeItem(`${str}name`)  // sessionStorage.removeItem(`${str}price`)  // sessionStorage.removeItem(`${str}picture`)  // }  function addToBean(event) {    let arr = document.getElementsByClassName("bean-button");    let index;    for (let i = 0; i < arr.length; i++) {      if (arr[i] === event.target) {        index = i;      }    }    let name = document.getElementsByClassName("type")[index].innerHTML;    let price = document.getElementsByClassName("cost")[index].innerHTML;    let picture = document.getElementsByTagName("img")[index].src;    let size = document.querySelector(".active");    console.log(size)    if(!size){      alert('Выберите размер');      return;    }    let flag = true,      str;    for (let i = 0; i < sessionStorage.length; i++) {      str = i;      if ((sessionStorage[`${str}name`] === name) && (sessionStorage[`${str}size`] === size.innerHTML)) {        //проверка что такого цвета нет в хранилище        alert("Товар уже в корзине");      //   deleteFromBean(str);        return;      }    }    if (flag) {      let leng = sessionStorage.length / 4 + 1;        for(let i=1;i<=leng;i++){          if(!sessionStorage.getItem(`${i}name`)){              sessionStorage.setItem(`${i}name`, name);              sessionStorage.setItem(`${i}price`, price);              sessionStorage.setItem(`${i}picture`, picture);              sessionStorage.setItem(`${i}size`, size.innerHTML);              break;          }        }    }  }  function addToLiked(event) {      let arr = document.getElementsByTagName("rect");      // console.log(arr);      // console.log(event.target)      let index;      for (let i = 0; i < arr.length; i++) {        if (arr[i] === event.target) {          index = i;        }      }      // console.log(index)        let name = document.getElementsByClassName("type")[index].innerHTML;      let price = document.getElementsByClassName("cost")[index].innerHTML;      let picture = document.getElementsByTagName("img")[index].src;      let size = document.querySelector(".active");      console.log(size)        if(!size){        alert('Выберите размер');        return;      }        let flag = true,        str;      for (let i = 0; i < sessionStorage.length; i++) {        str = i;        if ((sessionStorage[`${str}nameL`] === name) && (sessionStorage[`${str}sizeL`] === size.innerHTML)) {          //проверка что такого цвета нет в хранилище          alert("Товар уже в избранном");        //   deleteFromBean(str);          return;        }      }      if (flag) {        let leng = sessionStorage.length / 4 + 1;        for(let i=1;i<=leng;i++){          if(!sessionStorage.getItem(`${i}nameL`)){              sessionStorage.setItem(`${i}nameL`, name);              sessionStorage.setItem(`${i}priceL`, price);              sessionStorage.setItem(`${i}pictureL`, picture);              sessionStorage.setItem(`${i}sizeL`, size.innerHTML);              break;          }        }      }    } |

|  |
| --- |
| window.addEventListener("load", () => {      let temp = [];      let name, price, picture, item, size;      for (let i = 1; i < sessionStorage.length; i = i + 4) {        if (i == 1) {          if(!sessionStorage.getItem(`${i}nameL`)){              continue;          }          name = sessionStorage.getItem(`${i}nameL`);          price = sessionStorage.getItem(`${i}priceL`);          picture = sessionStorage.getItem(`${i}pictureL`);          size = sessionStorage.getItem(`${i}sizeL`);        } else {          if(!sessionStorage.getItem(`${i}nameL`)){              continue;          }          name = sessionStorage.getItem(`${parseInt(i + 3) / 4}nameL`);          price = sessionStorage.getItem(`${parseInt(i + 3) / 4}priceL`);          picture = sessionStorage.getItem(`${parseInt(i + 3) / 4}pictureL`);          size = sessionStorage.getItem(`${parseInt(i + 3) / 4}sizeL`);        }        item = { name: name, price: price, picture: picture, size: size };        temp.push(item);        console.log(temp)      }        const main = document.querySelector(".items-container");        temp.forEach((item) => {        const node = document.createElement("div");        node.classList.add("item");        //   const imageContainer = document.createElement("div");      //   imageContainer.classList.add("game-image");        const image = document.createElement("img");        image.src = item.picture;      //   imageContainer.appendChild(image);        node.appendChild(image);        let info =document.createElement("div")        info.classList.add("item-info")        let price=document.createElement("p");        price.innerHTML=`Цена: ${item.price}`;        let name=document.createElement("p");        name.innerHTML=item.name;        let size=document.createElement("p");        size.innerHTML=`Размер: ${item.size}`;        let toBean=document.createElement("button");        toBean.innerHTML="В корзину";        toBean.classList.add("bean-button")        info.appendChild(price)        info.appendChild(name)        info.appendChild(size)        info.appendChild(toBean)        node.appendChild(info)          let deleteButton=document.createElement("button")        deleteButton.innerHTML="X"        deleteButton.classList.add("deleet-btn")        node.appendChild(deleteButton)        //   deleteButton.addEventListener("click", deleteFromLiked)          main.appendChild(node);      });    });    //   let delButtons=Array.from(document.getElementsByClassName("del"))  //   delButtons.forEach((button)=>{button.addEventListener('click', deleteFromBean)})      //   function deleteFromLiked(event) {  //       console.log("s")  //       let arr = document.getElementsByClassName("del");  //       let index;  //       for (let i = 0; i < arr.length; i++) {  //         if (arr[i] === event.target) {  //           index = i;  //         }  //       }    //       let name = document.getElementsByClassName("game-title")[index].innerHTML;    //       let gamesContainersArray = Array.from(document.getElementsByClassName("game-container"));  //       let gamesArray = Array.from(document.getElementsByClassName("game-title"));  //       console.log(gamesArray)  //       let deleteIndex;  //       for (let i = 0; i < gamesArray.length; i++) {  //         if (gamesArray[i].innerHTML === name) {  //           deleteIndex = i;  //           console.log(gamesArray[i].innerHTML)  //         }  //       }  //       console.log(name)  //       console.log(deleteIndex)  //       console.log(gamesContainersArray)  //       gamesContainersArray[deleteIndex].parentNode.removeChild(gamesContainersArray[deleteIndex])  //       let str;  //       for (let i = 0; i < sessionStorage.length; i++) {  //           str = i;  //           if (sessionStorage[`${str}name`] === name) {  //               break;  //           }  //         }    //       console.log(str)  //     for(; str<sessionStorage.length/3; str++){  //       sessionStorage.setItem(`${str}name`, sessionStorage[`${str+1}name`] )  //     }  //   sessionStorage.removeItem(`${str}name`)  //   sessionStorage.removeItem(`${str}price`)  //   sessionStorage.removeItem(`${str}picture`)  //   }    function addToBean(event) {      let arr = document.getElementsByClassName("bean-button");      let index;      for (let i = 0; i < arr.length; i++) {        if (arr[i] === event.target) {          index = i;        }      }        let name = document.getElementsByClassName("type")[index].innerHTML;      let price = document.getElementsByClassName("cost")[index].innerHTML;      let picture = document.getElementsByTagName("img")[index].src;      let size = document.querySelector(".active");      console.log(size)        if(!size){        alert('Выберите размер');        return;      }        let flag = true,        str;      for (let i = 0; i < sessionStorage.length; i++) {        str = i;        if ((sessionStorage[`${str}name`] === name) && (sessionStorage[`${str}size`] === size.innerHTML)) {          //проверка что такого цвета нет в хранилище          alert("Товар уже в корзине");        //   deleteFromBean(str);          return;        }      }      if (flag) {        let leng = sessionStorage.length / 4 + 1;        sessionStorage.setItem(`${leng}name`, name);        sessionStorage.setItem(`${leng}price`, price);        sessionStorage.setItem(`${leng}picture`, picture);        sessionStorage.setItem(`${leng}size`, size.innerHTML);      }    } |

|  |
| --- |
| document.addEventListener("DOMContentLoaded", function() {      const decreaseButton = document.querySelector(".quantity-decrease");      const increaseButton = document.querySelector(".quantity-increase");      const quantityDisplay = document.querySelector(".quantity");      decreaseButton.addEventListener("click", function() {          let quantity = parseInt(quantityDisplay.textContent);          if (quantity > 1) {              quantity--;              quantityDisplay.textContent = quantity;          }      });      increaseButton.addEventListener("click", function() {          let quantity = parseInt(quantityDisplay.textContent);          quantity++;          quantityDisplay.textContent = quantity;      });  }); |

|  |
| --- |
| var xhr = new XMLHttpRequest();  xhr.open("GET", "about.xml", true);  xhr.setRequestHeader("Content-Type", "text/xml");  xhr.onreadystatechange = function() {      if (xhr.readyState === 4 && xhr.status === 200) {          var xml = xhr.responseXML;            var title = xml.querySelector("title").textContent;          var description = xml.querySelector("description").textContent;          var support = xml.querySelector("support").textContent;          var phone = xml.querySelector("phone").textContent;            var footerDiv = document.querySelector(".footer-section[data-section='4']");          footerDiv.innerHTML = "<h2>" + title + "</h2>" +                                "<p>" + description + "</p>" +                                "<p>" + support + "</p>" +                                "<p>" + phone + "</p>";      }  };  xhr.send(); |