**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**Факультет: Информационные технологии и управление**

**Кафедра: Компьютерная инженерия**

**Предмет: «Веб-системы и технологии»**

Курсовая работа

**Тема** «Разработка онлайновой информационно-справочной системы для интернет-магазина спортивной одежды»

**Группа:** 680.22

**Курc:** 3

**Специальность: 050616** Информационные технологии

**Студент**: Аббасов Мурад

**Руководитель:** **асс. Халилов М. Э.**

**Зав.кафедрой: доц.Рагимова Н.А.**

**Баку – 2025**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ**

**ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ**

**ФАКУЛЬТЕТ**: Информационные технологии и управление

**КАФЕДРА**: Компьютерная инженерия

**Группа** 680.22 **курс** III

**Специальность** **050616** Информационные технологии

**Студент** Аббасов Мурад

**Зачетная книжка**

**Руководитель курсовой работы**  **асс. Халилов М. Э.**

**Срок выдачи**

**Дата сдачи**

**Тема курсовой работы «Разработка онлайновой информационно-справочной системы для интернет-магазина спортивной одежды**»

**Отзыв руководителя курсовой**:

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

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

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

**Подпись студента**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись рук.курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись зав.кафедрой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата защиты курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Оценка**\_\_\_\_\_\_\_\_\_\_

**Пред.комиссии**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Члены комиссии**: 1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**СОДЕРЖАНИЕ**

0. **Введение………………………………………………………………………**

1.  **Теоритическая часть………...…………………………**

1.1 **Web-разработка………………………………………**

1.2 **HTML……………………………………………….**

1.3 **CSS****……………………………………………………**

1.4 **JavaScript………………………………………………………**

2. **Практическая часть…………………………………………**

2.1 **Цель курсовой работы и использованные технологии……...**

2.2 **Среда создания и структура проекта………………………………**

2.3 **Создание страниц, стилей и JS скриптов………………...**

2.3.1 **Создание main.css………………………………**

2.3.2 **Создание main.html страницы…………**

2.3.3 **Создание mensclothing.css…………**

2.3.4 **Создание mensclothing.html страницы…………**

2.3.5 **Создание womensclothing.css…………**

2.3.6 **Создание womensclothing.html страницы…………**

2.3.7 **Создание JavaScript скриптов…………….**

3. **Результат………………………………………………………………**

4.  **Литература………………………………………………………………**

**Введение**

Веб-технологии представляют собой совокупность инструментов, языков программирования и платформ, используемых для создания и поддержки веб-приложений, таких как интернет-магазины. С развитием интернета эти технологии эволюционировали, чтобы отвечать растущим требованиям пользователей, обеспечивая удобство, доступность и функциональность. Появление веб-технологий позволило широкому кругу людей, не обладающих глубокими знаниями в IT, активно использовать интернет для поиска информации, покупок и общения.

Ключевым фактором успеха современных веб-систем стал интуитивно понятный пользовательский интерфейс. Ранее сложные и громоздкие способы взаимодействия с компьютерами сдерживали внедрение новых технологий. Упрощение интерфейсов до уровня, понятного обычным пользователям, вызвало всплеск интереса к возможностям интернета и веб-приложений, включая онлайн-магазины.

Веб-технологии, обеспечивающие работу Всемирной паутины (World Wide Web, WWW), делятся на две основные категории:

1. **Клиентские технологии**: Отвечают за создание и отображение веб-страниц в браузере. К ним относятся:
   * **HTML** — язык гипертекстовой разметки для создания структуры страниц.
   * **CSS** — для оформления и стилизации страниц.
   * **JavaScript** — язык программирования (скриптинга) на котором создаются все динамические (интерактивные) элементы страниц.
2. **Серверные технологии**: Обеспечивают обработку данных и взаимодействие с базами. Примеры включают PHP, Perl и системы управления базами данных (SQL), которые поддерживают хранение информации о товарах, заказах и пользователях.

Работа веб-приложений основана на архитектуре "клиент-сервер". Клиент (браузер пользователя) отправляет запрос на сервер, который обрабатывает его и возвращает ответ, содержащий данные (например, страницу с каталогом товаров) или сообщение об ошибке. Эта двухзвенная модель лежит в основе большинства интернет-сервисов, включая онлайновые информационно-справочные системы, такие как интернет-магазин спортивной одежды.

В рамках данной курсовой работы акцент сделан только на клиентскую часть.

Курсовая работа посвящена разработке онлайновой информационно-справочной системы для интернет-магазина, специализирующегося на продаже спортивной одежды. Основная цель — создать удобный, безопасный и интуитивно понятный веб-сайт, ориентированный на потребности покупателей.

Процесс разработки включает следующие ключевые этапы:

1. **Анализ требований**: На первом этапе проводится сбор информации о потребностях заказчика. Определяются ассортимент товаров (спортивная одежда), целевая аудитория (спортсмены) и желаемый функционал сайта, такой как каталог товаров, поиск, отзывы.
2. **Проектирование интерфейса**: На этом этапе создаются прототипы страниц, выбираются цветовая палитра, шрифты и общая стилистика. Особое внимание уделяется удобству навигации и адаптивности дизайна для разных устройств.
3. **Кодирование сайта**: Реализация проекта осуществляется с использованием современных веб-технологий: HTML для структуры, CSS для оформления и JavaScript для интерактивных элементов, таких как поиск товаров и перевод страницы.
4. **Тестирование**: После завершения разработки сайт проверяется на корректность работы в популярных браузерах (Chrome, Firefox, Safari) и на различных устройствах (ПК, планшеты, смартфоны). Это гарантирует стабильность и доступность системы.

Интернет-магазин разрабатывается с акцентом на удобство пользователя. Сайт должен обеспечивать:

* Быстрый и простой просмотр каталога товаров.
* Удобный поиск по категориям, брендам или ключевым словам.
* Интуитивно понятный процесс оформления заказа.

1. **Теоритическая часть**

**1.1** **Web-разработка**

**Web-разработка** объединяет технологии для создания веб-сайтов и приложений, работающих через интернет. Она основана на клиент-серверной архитектуре, где клиентские языки отображают страницы в браузере, а серверные обрабатывают данные.

**Клиенты** являются обычными пользователями, подключёнными к Интернету посредством устройств (например, компьютер подключён к Wi-Fi, или ваш телефон подключён к мобильной сети) и программного обеспечения, доступного на этих устройствах (как правило, браузер, например, Firefox или Chrome).

**Серверы** - это компьютеры, которые хранят веб-страницы, сайты или приложения. Когда клиентское устройство пытается получить доступ к веб-странице, копия страницы загружается с сервера на клиентский компьютер для отображения в браузере пользователя.

Клиентские языки, такие как HTML, CSS и JavaScript, обеспечивают функциональность веб-страниц в браузере. Они программируют поведение страниц, работают локально и не требуют постоянного взаимодействия с сервером, но зависят от возможностей браузера.

**JavaScript** — один из трех языков, которые должны знать все веб-разработчики:

1. **HTML** для определения содержимого веб-страниц,
2. **CSS** для задания макета и стиля веб-страниц,
3. **JavaScript** — сценарный или программирующий язык, который позволяет реализовывать сложные функции на веб-страницах.

**1.2****HTML**

**HTML (HyperText Markup Language - язык гипертекстовой разметки)** не является языком программирования; это язык разметки, используемый для определения структуры веб-страниц, посещаемых пользователями. Они могут иметь сложную или простую структуру, всё зависит от замысла и желания веб-разработчика. HTML состоит из ряда элементов, которые вы используете для того, чтобы охватить, обернуть или разметить различные части содержимого, чтобы оно имело определённый вид или срабатывало определённым способом.

HTML использует теги и аргументы к ним для построения страниц.

Теги обозначаются через символы < и >, а также закрываются добавлением символа / к ним. Например, <p> открывает тег для создания нового параграфа, а </p> его закрывает. Не все теги требуют закрытия, например тег обрывания линии (breake line) <br> не требует </br>.

Атрибуты в HTML задают дополнительные параметры тегам, например стиль CSS через style="…" (inline-css) или class=".example".

Опишем назначение большинства тегов HTML (в алфавитном порядк"е):

1. **<a>** используется для создания гиперссылки на внешнюю страницу (сайт)
2. **<abbr>** определяет аббревиатуру или акроним и выделяет их
3. **<article>** определяет независимый, самодостаточный контент, который иметь смысл сам по себе, и его можно распространять независимо от остальной части сайта
4. **<aside>** определяет некоторый контент, который лишь косвенно связан с окружающим его контентом
5. **<b>** делает текст жирным, без каких-либо других атрибутов
6. **<bdi>** используется, когда текст содержит и языки, которые пишутся слева направо, и языки, которые пишутся справа налево
7. **<bdo>** переназначает направление, в котором пишется язык
8. **<blockquote>** указывает цитату из другого источника в отдельном блоке
9. **<body>** содержит главное содержимое HTML-документа, например заголовки, абзацы, изображения, гиперссылки и т.д. В HTML-документе может быть только один <body>
10. **<br>** делает перенос строки (не требует </br>)
11. **<caption>** определяет заголовок таблицы, который будет выведен над ней
12. **<cite>** определяет название творческой работы (например, книги, стихотворения, песни, фильма, картины, скульптуры и т. д.). Обычно отображается курсивом
13. **<code>** используется для определения фрагмента компьютерного кода. Отображается в моноширинном шрифте браузера
14. **<dd>** определяет описание в <dl> листе, часто используется под <dt> элементом
15. **<del>** определяет текст, который был удален из документа (зачеркнутый)
16. **<details>** указывает дополнительные сведения, которые пользователь может открывать и закрывать (спойлер)
17. **<dfn>** выделяет термин, используемый в тексте (курсивом)
18. **<dialog>** определяет диалоговое окно или подокно
19. **<div>** используется как контейнер для элементов HTML, которые затем стилизуются с помощью CSS или обрабатываются с помощью JavaScript
20. **<dl>** определяет список описаний. <dl> используется вместе с <dt> и <dd>
21. **<dt>** определяет имя в списке описаний. <dt> используется вместе с <dl> и <dd>
22. **<em>** акцентирует какой-то отрывок текста или слово из него (курсивом)
23. **<footer>** используется для выделения информации об авторстве, информации об авторских правах, контактную информацию и т.д.
24. **<form>** создает форму, которую пользователь может заполнять. Для заполнения формы используется поле ввода <input>
25. **<h1>**, **<h2>**, ... , **<h6>** используются для определения заголовков. <h1> яаляется самым "важным" (большим) заголовком, а <h6> — наименее важным (мелким)
26. **<head>** является контейнером для метаданных и размещается между тегом <html> и тегом <body>
27. **<header>** — контейнер, который может содержать заголовки, значок, авторство
28. **<hgroup>** используется для окружения заголовка и связанных <p> элементов
29. **<hr>** Тег определяет тематический разрыв на странице (разделение линией)
30. **<html>** представляет собой корень HTML-документа и является контейнером для всех остальных элементов HTML (кроме <!DOCTYPE>)
31. **<i>** выделяет часть текста курсивом
32. **<input>** определяет поле ввода, в которое пользователь может вводить данные
33. **<ins>** определяет текст, вставленный в документ (подчеркивает текст)
34. **<kbd>** используется для определения клавиш или комбинаций клавиш, вводимых с клавиатуры. Содержимое внутри отображается в моноширинном шрифте
35. **<li>** определяет элемент списка. <li> используется внутри упорядоченных (<ol>) и неупорядоченных списков (<ul>), а также в списках меню (<menu>)
36. **<menu>** создает неупорядоченный список. <menu> является альтернативой тегу <ul>, и браузеры будут обрабатывать эти два списка одинаково.
37. **<ol>** определяет упорядоченный список. Может быть числовым или алфавитным. Используется совместно с <li> для определения элементов списка
38. **<option>** определяет опцию в выпадающем списке <select>, которую пользователь может выбрать
39. **<p>** определяет абзац. Браузеры автоматически добавляют одну пустую строку до и после каждого элемента <p>
40. **<q>** создает краткую цитату, выделенную кавычками (в отличие от <blockquote>, не создает отдельный блок — цитата добавляется в текст без разделений)
41. **<s>** указывает текст, который больше не является правильным и/или актуальным. Текст будет отображаться зачеркнутым. <s> схож с <del>, но их не надо путать
42. **<section>** определяет раздел в документе. <section> может использоваться с <h1>...<h6> и <p>
43. **<select>** используется для создания выпадающего списка, используется вместе с <option> и часто задается в <form>. Позволяет пользователю выбрать одну из опций выпадающего списка
44. **<strong>** используется для определения текста высокой важности. Содержимое отображается жирным шрифтом. Не следует путать с <b>
45. **<style>** определяет информацию о стиле HTML документа с помощью CSS. Элемент <style> должен располагаться в разделе <head>
46. **<sub>** определяет подстрочный текст, который отображается на полсимвола ниже обычной строки. Может отображается шрифтом меньшего размера
47. **<sup>** определяет надстрочный текст, который отображается на полсимвола выше обычной строки. Может отображается шрифтом меньшего размера
48. **<table>** тег определяет HTML-таблицу. HTML-таблица состоит из одного элемента <table> и одного или нескольких элементов <td>, <th> и <tr>
49. **<td>** определяет стандартную ячейку данных в HTML-таблице. Должен находиться внутри <tr>
50. **<th>** определяет ячейку заголовка в HTML-таблице. Должен находиться внутри <tr>
51. **<tr>** определяет строку в таблице HTML-таблице. Содержит один или несколько элементов <th> или <td>
52. **<title>** определяет заголовок документа. Заголовок должен быть только один и содержать только текст. Поисковые системы как Google, Yandex, Bing и другие используют заголовки для определения порядка результатов
53. **<u>** подчеркивает текст, который оформлен иначе, чем обычный текст (например, слова с ошибками). Не путать с <a>, <b> и <strong>

Примеры атрибутов HTML:

1. **id — п**рисваивает элементу уникальный идентификатор на странице, который используется для обращения к элементу через CSS или JavaScript. **Пример**:

<div id="header">

*Значение должно быть уникальным в пределах страницы.*

1. **class — у**казывает класс или группу классов для стилизации элемента через CSS или взаимодействия с JavaScript. **Пример**:

<p class="text-bold">

*Один элемент может иметь несколько классов, разделенных пробелом.*

1. **src — у**казывает источник внешнего ресурса, например, изображения или скрипта. **Пример**:

<img src="image.jpg">

*Используется с тегами <img>, <script>, <iframe> и др.*

1. **alt — з**адает альтернативный текст для элемента, который отображается, если ресурс недоступен, или используется скринридерами для доступности. **Пример**:

<img src="image.jpg" alt="Описание изображения">

*Обязателен для тега <img> с точки зрения стандартов доступности.*

1. **href — у**казывает адрес ресурса, на который ведет гиперссылка. **Пример**:

<a href="page.html">

Используется с тегом <a> и может указывать на внешние или внутренние ресурсы.

С остальными атрибутами можно ознакомиться в документации HTML.

**1.3****CSS**

**CSS (Cascading Style Sheets)** — это технология, используемая для оформления и стилизации веб-страниц, созданных с помощью HTML. CSS позволяет задавать внешний вид элементов, таких как цвета, шрифты, размеры, расположение и анимации, отделяя дизайн от структуры документа. Это повышает гибкость и удобство разработки.

### Как работает CSS

Когда браузер отображает документ, он должен совместить его содержимое с его стилями. Этот процесс идёт в несколько этапов, о которых мы сейчас поговорим. Держите в уме, что это очень упрощённая версия того как браузер действительно загружает веб-страницу, а также то, что разные браузеры делают это по разному. Но, происходит, грубо говоря, следующее:

1. Браузер получает HTML-страницу (например, из Интернета)
2. Преобразует HTML в DOM (Document Object Model). DOM (или DOM-дерево) - это представление страницы в памяти компьютера. Подробнее на DOM мы остановимся чуть позже.
3. Затем браузер забирает все ресурсы и описания, связанные с HTML-документом, например: встроенные картинки, видео и стили CSS.
4. После этого браузер анализирует полученный CSS код, сортирует описанные там правила в зависимости от их селекторов и раскладывает их в различные «корзины»: элементы, классы, идентификаторы(ID) и т.п. Основываясь на найденных селекторах браузер понимает какие правила относятся к определённым «узлам» в DOM-дереве и применяет их по мере необходимости (этот промежуточный шаг называют «формированием дерева представления» или «формированием дерева рендеринга»)
5. Дерево представления (render tree) формируется в том порядке, в каком оно затем должно будет отображаться, когда все правила будут применены.
6. Затем происходит визуальное отображение контента на странице (этот этап называется «отрисовкой»)

### Структура CSS

CSS состоит из правил, каждое из которых включает:

* **Селектор**: Указывает, к каким элементам HTML применяется стиль (например, p для абзацев, .class для классов, #id для идентификаторов).
* **Свойства и значения**: Определяют, как будет выглядеть элемент (например, color: blue; задает синий цвет текста).

Пример правила:

h1 {

font-size: 24px;

color: #333;

}

Здесь селектор h1 применяет стили к заголовкам первого уровня, задавая размер шрифта и цвет.

### Способы подключения CSS

1. **Внешний файл**: Стили хранятся в отдельном файле с расширением .css и подключаются через тег <link> в HTML.
2. **Внутренние стили**: Указываются внутри тега <style> в секции <head> документа.
3. **Встроенные стили**: Задаются через атрибут style непосредственно в HTML-теге.

### Основные свойства CSS

1. **color**: Задает цвет текста (например, color: red;).
2. **font-family**: Определяет шрифт (например, font-family: Arial, sans-serif;).
3. **margin** и **padding**: Управляют внешними и внутренними отступами.
4. **display**: Контролирует тип отображения элемента (например, block, inline, flex).
5. **background**: Задает фон (цвет, изображение, например, background: #f0f0f0;).

CSS является неотъемлемой частью веб-разработки, обеспечивая визуальную привлекательность и адаптивность страниц. Благодаря модульности и гибкости, CSS позволяет создавать современные интерфейсы, соответствующие стандартам дизайна.

***1.4 JavaScript***

**JavaScript** (**JS**) — это легковесный интерпретируемый (или JIT-компилируемый) язык программирования с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и в других программных продуктах, например, Node.js или Apache CouchDB. JavaScript это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования.

Стандартом языка JavaScript является ECMAScript. По состоянию на 2012 год, все современные браузеры полностью поддерживают ECMAScript 5.1. Старые версии браузеров поддерживают по крайней мере - ECMAScript 3. 17 июня 2015 года состоялся выпуск шестой версии ECMAScript. Эта версия официально называется ECMAScript 2015, которую чаще всего называют ECMAScript 2015 или просто ES2015. С недавнего времени стандарты ECMAScript выпускаются ежегодно.

С полным синтаксисом JavaScript можно ознакомиться в оффициальной документации, ниже будут приведены некоторые осообенности.

В JavaScript существует три вида объявлений:

1. var объявляет переменную, инициализация переменной значением является необязательной.
2. let объявляет локальную переменную в области видимости блока, инициализация переменной значением является необязательной.
3. const объявляет именованную константу, доступную только для чтения.

JavaScript имеет шесть типов данных, которые являются примитивами:

* 1. **Boolean**. **true** и **false**
  2. **null**. Специальное ключевое слово, обозначающее нулевое или «пустое» значение. Поскольку JavaScript чувствителен к регистру, null не то же самое, что Null, NULL или любой другой вариант.
  3. **undefined**. Свойство глобального объекта; переменная, не имеющая присвоенного значения, обладает типом undefined.
  4. **Number**. 42 или 3.14159.
  5. **String**. "Howdy".
  6. **Symbol** (ECMAScript 6)

И тип **Object**, не являющийся примитивом.

JavaScript — это динамически типизированный язык. Это означает, что вам не нужно указывать тип данных переменной, когда вы её объявляете, типы данных преобразуются автоматически по мере необходимости во время выполнения скрипта.

**2. Практическая часть**

**2.1** **Цель курсовой работы и использованные технологии**

Целью данной курсовой работы является разработка онлайновой информационно-справочной системы для интернет-магазина спортивной одежды.

Для этой цели были использованы ранее описанные технологии:

* HTML
* CSS
* JavaScript

**2.2** **Среда создания и структура проекта**

Редактором для данной курсовой послужил **Kate**, текстовой редактор созданный разработчиками **KDE**.

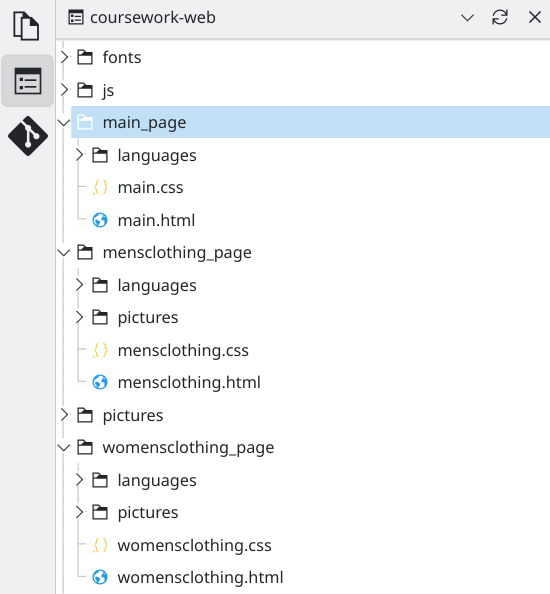
Все файлы и директории, используемые в создаваемом сайте хранятся в одной директории **coursework-web/** (корневая директория проекта).

Сайт будет состоять из 3 страниц:

1. main.html в директории main\_page/
2. mensclothing.html в директории mensclothing\_page/
3. womensclothing.html в директории womensclothing\_page/

Внутри каждой под-директории этих страниц имеются такие директории как languages/ и pictures/ и файл .css для стилей CSS. Также в корневой директории имеются директории js/, pictures/ и fonts/, хранящие общие JavaScript скрипты, картинки и шрифты соответственно.

Готовую структуру проекта можно наблюдать на **рисунке 1**.



**Рисунок 1. Структура проекта**

**2.3** **Создание страниц, стилей и JS скриптов**

**2.3.1 Создание main.css**

Для начала создадим заранее **main.css** стиль для нашей будующей **main.html** страницы.

Для начала добавим шрифт, используемый на странице:

**@font-face {**

**/\* https://fontesk.com/onest-typeface \*/**

**font-family: "OnestRegular";**

**src: url("../fonts/OnestRegular1602-hint.ttf");**

**}**

, указав имя семейства шрифтов в **font-family** и ссылку на нее в

**src: url("…")**.

Далее зададим начальный стиль для всех элементов:

**\*, html {**

**margin:0;**

**padding:0;**

**box-sizing:border-box;**

**flex-direction: column;**

**font-family: "OnestRegular";**

**word-wrap: break-word;**

**}**

, где:

* **margin: 0** задает нулевой отступ,
* **padding: 0** задает внутренний нулевой отступ,
* **box-sizing:border-box** говорит браузеру учитывать любые границы и внутренние отступы в ширине и высоте элемента,
* **flex-direction: column** распологает элементы в столбец сверху вниз,
* **font-family: "OnestRegular"** задает шрифт,
* **word-wrap: break-word** переносит слова целиком на другую строку.

Задаем высоту в 100% от страницы и нулевой отступ у **html** и **body**:

**html, body {**

**height: 100%;**

**margin: 0;**

**}**

Убираем жирный текст у **h1** … **h6**:

**h1, h2, h3, h4, h5, h6 {**

**font-weight: normal;**

**}**

Задаем отступ 2 пикселя со всех сторон у элемента **p** *(параграф)*:

**p {**

**margin: 2px;**

**}**

Задаем цвет фона у body:

**body {**

**background: #f57971;**

**}**

Добавляем стиль для верхнего баннера, задав отступы, ширину, цвета фона и текста и т.д:

**.topbanner\_div {**

**margin-bottom: 20px;**

**margin-left: 0px;**

**margin-right: 0px;**

**width: 100%;**

**background-color: #bf564e;**

**color: white;**

**text-align: center;**

**padding-top: 10px;**

**padding-bottom: 10px;**

**padding-left: 10px;**

**padding-right: 10px;**

**}**

Добавляем стиль для элемента select, добавив скругленную полупрозрачную обводку, отступы, цвет и т.д:

**select {**

**border: 4px solid rgba(207, 102, 95, .8);**

**border-radius: 40px;**

**margin-top: 5px;**

**margin-bottom: 5px;**

**margin-left: auto;**

**margin-right: auto;**

**background-color: #fa827a;**

**text-align: left;**

**padding: 10px;**

**}**

Добавим стиль для специального блока, задающего структуру нашего body:

**.flex\_div {**

**min-height: 100%;**

**display: flex;**

**}**

, где **display: flex** позволяет элементам страницы автоматически распределяться по пустому пространству страницы.

Добавим стиль для блока, служащего как ссылка на **mensclothing.html**:

**.men\_div {**

**border: 4px solid #D0F0F6;**

**border-radius: 40px;**

**margin-top: 5px;**

**margin-bottom: 5px;**

**margin-left: auto;**

**margin-right: auto;**

**width: 95%;**

**background-color: #82DCED;**

**text-align: left;**

**padding-left: 20px;**

**padding-right: 20px;**

**padding-top: 35px;**

**padding-bottom: 35px;**

**}**

И аналогичный для ссылки на **womensclothing.html**:

**.women\_div {**

**border: 4px solid #FFE5EC;**

**border-radius: 40px;**

**margin-top: 5px;**

**margin-bottom: 5px;**

**margin-left: auto;**

**margin-right: auto;**

**width: 95%;**

**background-color: #F0A1B6;**

**text-align: left;**

**padding-left: 20px;**

**padding-right: 20px;**

**padding-top: 35px;**

**padding-bottom: 35px;**

**}**

И в конце добавим стиль для нижнего копилефт блока страницы:

**.copyleft\_div {**

**margin: 0px;**

**margin-top: 50px;**

**margin-bottom: 0px;**

**width: 100%;**

**background-color: #bf564e;**

**color: white;**

**text-align: left;**

**padding: 0px;**

**padding-bottom: 10px;**

**}**

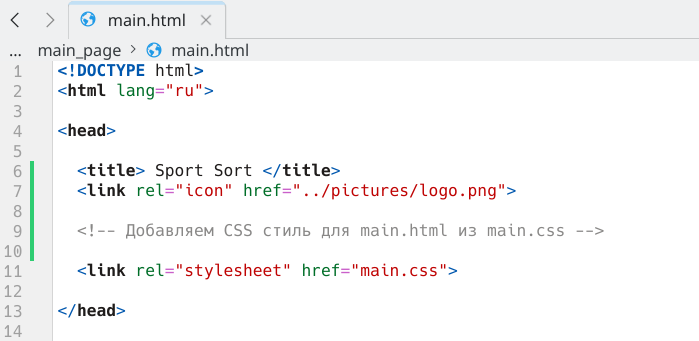
**2.3.2** **Создание main.html страницы**

В этой части будет описано пошаговое создание main.html страницы, т.е. главной страницы курсовой работы. *На рисунках, отображающих структуру страницы содержатся комментарии к коду, но в самой курсовой они опущены.*

Далее добавим в main.html:

1. **<!DOCTYPE>**, указывающий что данный документ является документом HTML;
2. **<html lang="ru">**, открывающий главный тег HTML (с атрибутом lang для указания языка страницы), в который войдет вся наша страница;
3. **<head>**, открывающий часть HTML кода, который содержит метадату (заголовок страницы, стили и т.д.);
4. **<title> Sport Sort </title>,** указывающий заголовок нашей страницы;
5. **<link rel="icon" href="../pictures/logo.png">** через который мы задаем иконку нашей страницы;
6. **<link rel="stylesheet" href="main.css">**, связывающий наш main.css с нашим main.html, где атрибут **href** указывает путь к нашему файлу стилей,а **rel** указывает его отношение к main.html;
7. **</head>**, закрывающий тег <head>.

(рисунок 2)



**Рисунок 2**

Далее мы создаем <body>, в котором будет основная структура нашей страницы:

Добавляем внешние скрипты JS с помощью тега script:

1. **<script src="../js/changelang.js"> </script>**, скрипт меняющий язык нашей страницы,
2. **<script src="../js/resizeSelect.js"> </script>**, скрипт изменяющий размер элемента (тега) **select**, в зависимости от длины текста.

*(Так как данные скрипты являются общими, их код будет описан позже, после описания остальных страниц)*

Ниже будут созданы элементы, имеющие определенные классы CSS стилей. Эти классы будут описаны позже, в создании файла **main.css**.

Открываем специальный блок, который поможет создать правильную структуру нашей страницы *(а именно* *— возможность распределения блоков в самом верху или низу страницы)* c атрибутом **class****="flex\_div"**:

**<div** **class****="flex\_div">**

В нем мы создадим другой блок — верхний баннер нашей страницы:

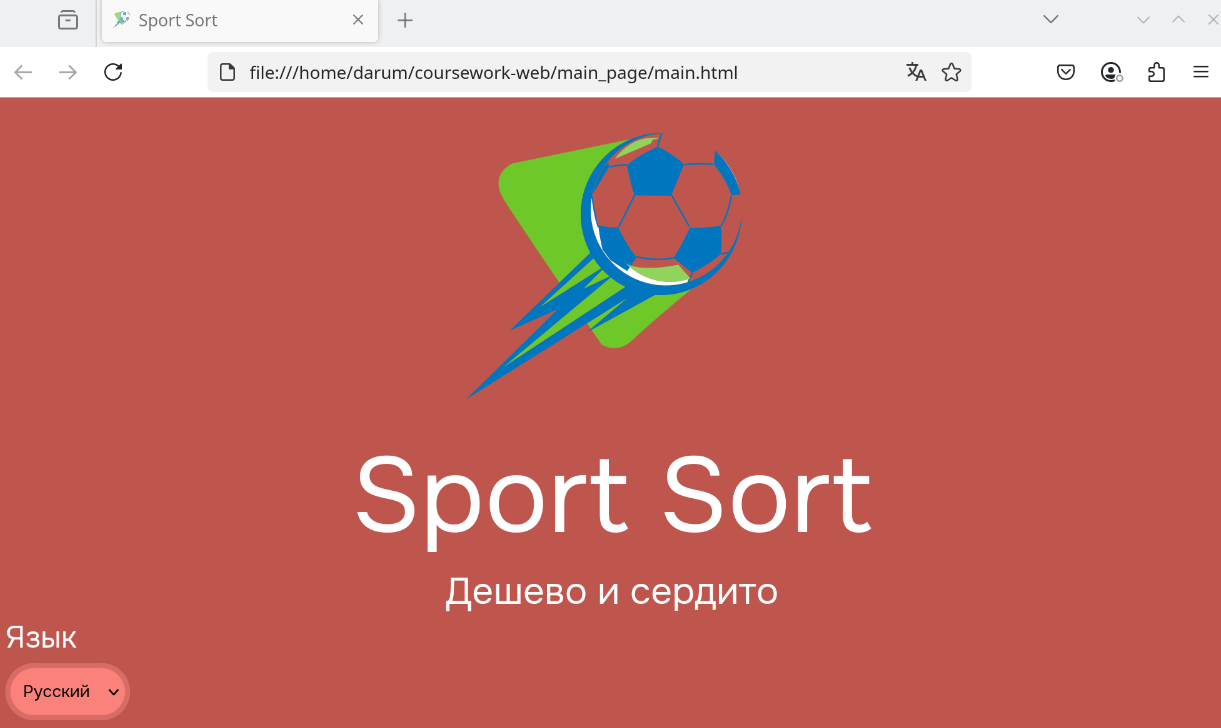
1. **<div class="topbanner\_div">**, открывающий блок баннера;
2. **<img src="../pictures/logo.png" alt="LOGO" width="250">** для создания лого нашего сайта, где **alt** указывает текст, показываемый при ошибке загрузки изображения, а **width** задает ширину;
3. **<h1 style=****"font-size: 85px"> Sport Sort </h1>** —добавляем название сайта, задав максимальную "важность" через тег **h1**;
4. **<h2** **style="font-size: 30px">** **<p lang-attr=topbanner1> Дешево и сердито </p> </h2>** — добавляем слоган сайта через h2, где:
   * **style="font-size: 30px"** — задает размер шрифта,
   * **<p** **lang-attr=topbanner1>** — создает параграф с кастомным атрибутом lang-attr для перевода через наш JS скрипт,
5. **<div style="text-align: left">** — открываем отдельный блок, использующий inline css атрибут **style="text-align: left"** *(размещение текста слева)*;
6. **<h2 lang-attr=langcur style="****margin: 0px">Язык</h2>**, где:
   * **margin: 0px** указывает, что текст будет иметь отступт 0 пикселей со всех сторон;
7. **<select** **id="langSelect"** **onchange="changeLang(); resizeSelect(this)">** — создаем элемент select, в котором пользователь сможет задать язык страницы, где:
   * **id="langSelect"** указывает уникальный индификатор, используемый в скрипте,
   * **onchange="****changeLang(); resizeSelect(this)**" выполняет две функции, **changeLang()** и **resizeSelect(this)**, при каждой смене выбранного варианта в **select**.
8. **<option value="ru" selected>Русский</option>** — вариант элемента **select**, обозначающий русский (стандартный) язык страницы;
9. **<option value="en">English</option>** — вариант элемента **select**, обозначающий английский язык страницы;
10. **</select>** — закрываем тег **<select>**;
11. **</div>** — закрываем под-блок баннера;
12. **</div>** — закрываем наш блок верхнего баннера.

(рисунок 3)

****

***Рисунок 3***

Полученный баннер можно рассмотреть на **рисунке 4**:

****

**Рисунок 4**

Далее мы создаем отдельный <div> в котором будут находиться ссылки на остальные две страницы проекта. В него мы добавляем ссылку на mansclothing.html:

1. **<div class="empty\_div" style="margin: 60px">** — пустой блок;
2. **</div>**
3. **<a** **href="../mensclothing\_page/mensclothing.html"** **style="text-decoration:none">** — ссылку на mensclothing.html, где:

* **href="../mensclothing\_page/mensclothing.html"** содержит саму ссылку,
* **style="text-decoration:none"** убирает декорации текста *(нижнее подчеркивание ссылки)*

1. **<div class="men\_div">** — блок, задающий стиль нашей ссылки;
2. **<h1> <b lang-attr=mensclothing>Мужской отдел</b> </h1>** — текст высокой важности с атрибутом для перевода;
3. **<p><h3 lang-attr=mensportcl>Мужская спортивная одежда</h3></p>** — текст с атрибутом для перевода;
4. **</div>** — закрываем блок;
5. **</a>** — закрываем ссылку *(так весь наш блок является ссылкой)*.

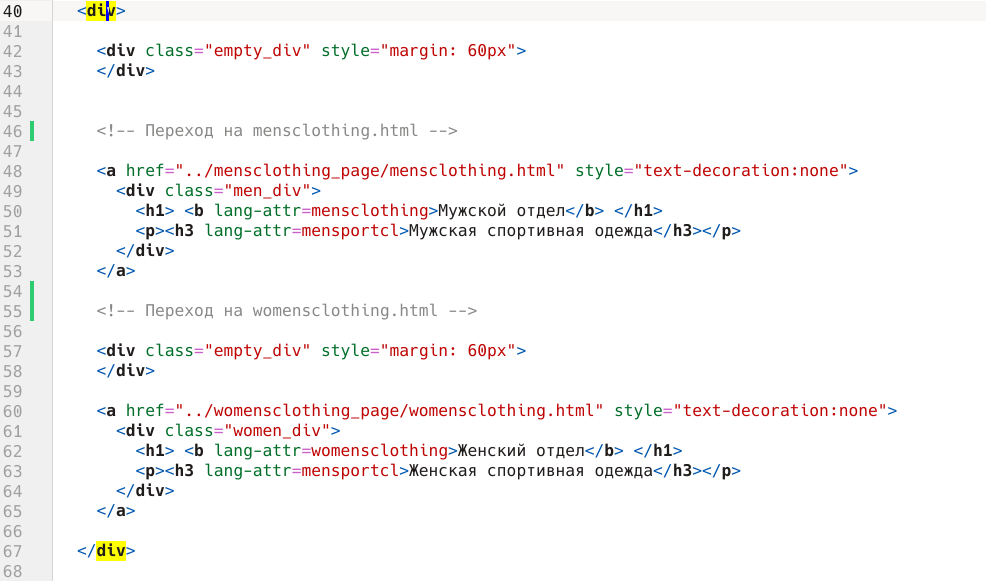
Далее добавляем еще один разделительный пустой блок:

1. **<div class="empty\_div" style="margin: 60px">**
2. **</div>**

И создаем аналогичную ссылку для страницы womensclothing.html:

1. **<a href="../womensclothing\_page/womensclothing.html" style="text-decoration:none">**
2. **<div class="women\_div">**
3. **<h1> <b lang-attr=womensclothing>Женский отдел</b> </h1>**
4. **<p><h3 lang-attr=mensportcl>Женская спортивная одежда</h3></p>**
5. **</div>**
6. **</a>**

*(рисунок 5)*

**

**Рисунок 5**

Полученный результат *(рисунок 6)*:



**Рисунок 6**

Наконец, добавляем нижний баннер нашей страницы:

1. **<div class="copyleft\_div">** — добавляем блок c классом **copyleft\_div**, который будет содержать наш копилефт;
2. <h1> <b style="font-size: 85px"> 🄯 </b> copyleft <h1 lang-attr=allrights> все права защищены </h1> </h1> — добавляем текст копилефта, содержащий:

* **<b style="font-size: 85px">** — тег **<b>** который делает текст жирным с атрибутом задающим размер текста 85 пикселей
* и уже описанные ранее теги;

1. **</div>** — закрываем блок.

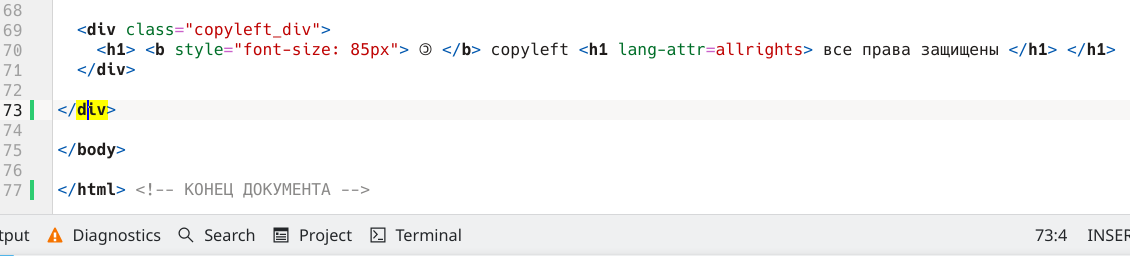
Под конец, закрываем <div class="flex\_div">, <body> и <html>:

</div>

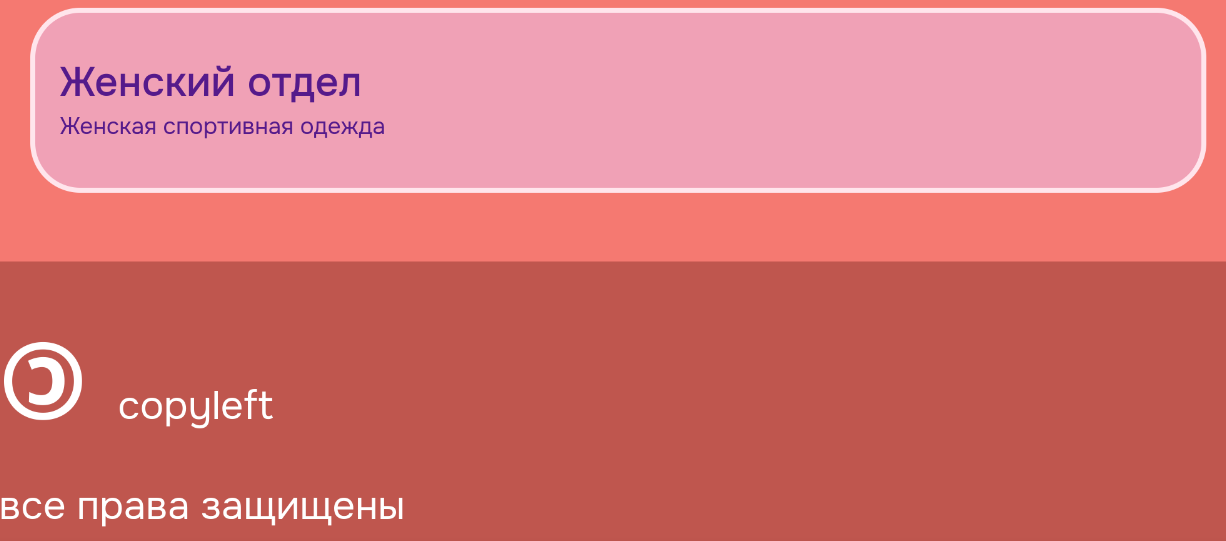
</body>

</html>

*(рисунок 7)*

**Рисунок 7**

Копилефт баннер можно рассмотреть на **рисунке 8**:

**Рисунок 8**

**2.3.3 Создание** **mensclothing.css**

Для начала создадим заранее **mensclothing.css** стиль для нашей будующей **mensclothing.html** страницы.

Для начала добавим шрифт, используемый на странице:

**@font-face {**

**/\* https://fontesk.com/onest-typeface \*/**

**font-family: "OnestRegular";**

**src: url("../fonts/OnestRegular1602-hint.ttf");**

**}**

, указав имя семейства шрифтов в **font-family** и ссылку на нее в

**src: url("…")**.

Далее зададим начальный стиль для всех элементов:

**\*, html {**

**margin:0;**

**padding:0;**

**box-sizing:border-box;**

**flex-direction: column;**

**font-family: "OnestRegular";**

**word-wrap: break-word;**

**}**

, где:

* **margin: 0** задает нулевой отступ,
* **padding: 0** задает внутренний нулевой отступ,
* **box-sizing:border-box** говорит браузеру учитывать любые границы и внутренние отступы в ширине и высоте элемента,
* **flex-direction: column** распологает элементы в столбец сверху вниз,
* **font-family: "OnestRegular"** задает шрифт,
* **word-wrap: break-word** переносит слова целиком на другую строку.

Задаем высоту в 100% от страницы и нулевой отступ у **html** и **body**:

**html, body {**

**height: 100%;**

**margin: 0;**

**}**

Убираем жирный текст у **h1** … **h6**:

**h1, h2, h3, h4, h5, h6 {**

**font-weight: normal;**

**}**

Задаем отступ 2 пикселя со всех сторон у элемента **p** *(параграф)*:

**p {**

**margin: 2px;**

**}**

Задаем цвет фона у body:

**body {**

**background: #f57971;**

**}**

Добавляем стиль для верхнего баннера, задав отступы, ширину, цвета фона и текста и т.д:

**.topbanner\_div {**

**margin-bottom: 20px;**

**margin-left: 0px;**

**margin-right: 0px;**

**width: 100%;**

**background-color: #bf564e;**

**color: white;**

**text-align: center;**

**padding-top: 10px;**

**padding-bottom: 10px;**

**padding-left: 10px;**

**padding-right: 10px;**

**}**

Добавляем стиль для элемента select, добавив скругленную полупрозрачную обводку, отступы, цвет и т.д:

**select {**

**border: 4px solid rgba(207, 102, 95, .8);**

**border-radius: 40px;**

**margin-top: 5px;**

**margin-bottom: 5px;**

**margin-left: auto;**

**margin-right: auto;**

**background-color: #fa827a;**

**text-align: left;**

**padding: 10px;**

**}**

Добавляем стиль для строки поиска **input**:

**input {**

**border: 4px solid rgba(207, 102, 95, .8);**

**border-radius: 40px;**

**margin-top: 5px;**

**margin-bottom: 5px;**

**margin-left: auto;**

**margin-right: auto;**

**background-color: #fa827a;**

**text-align: left;**

**font-size: 20px;**

**padding: 10px;**

**padding-right: 50%;**

**outline: none;**

**color: rgba(255, 255, 255, .8);**

**}**

Добавим стиль для специального блока, задающего структуру нашего body:

**.flex\_div {**

**min-height: 100%;**

**display: flex;**

**}**

, где **display: flex** позволяет элементам страницы автоматически распределяться по пустому пространству страницы.

Добавляем стиль для разделительного блока:

**.spacer\_div {**

**flex: 1;**

**}**

, где **flex** свойство, определяющее способность гибкого элемента растягиваться или сжиматься для заполнения собой доступного свободного пространства.

Далее добавляем стиль для блока-сетки, в котором будут распологаться товары:

**.grid\_div {**

**border: 10px solid #D0F0F6;**

**margin-top: 50px;**

**display: grid;**

**background-color: rgba(130, 220, 237, .5);**

**padding: 30px;**

**border-radius: 40px;**

**grid-template-columns: repeat(auto-fit, minmax(350px, 1fr));**

**gap: 16px;**

**padding: 16px;**

**}**

Добавляем стиль, который будет передаваться всем элементам **div** в **grid\_div** классе:

**.grid\_div > div {**

**border-radius: 15px;**

**background-color: #82DCED;**

**border: 5px solid #D0F0F6;**

**padding: 3px;**

**margin: 5px;**

**font-size: 30px;**

**text-align: center;**

**align-self: flex-end;**

**color: white;**

**}**

И создаем стиль внутреннего блока для блоков нашей сетки:

**.gridinside\_div {**

**background-color: rgba(208, 240, 246, .8);**

**padding: 2px;**

**border-radius: 7px;**

**align-self: flex-end;**

**}**

Создаем блок в котором будут отзывы на товары:

**.feedback\_div {**

**margin: 0px;**

**margin-top: 50px;**

**margin-bottom: 0px;**

**margin-left: 40px;**

**margin-right: 40px;**

**border-radius: 30px;**

**width: auto;**

**background-color: #bf564e;**

**color: white;**

**text-align: left;**

**padding: 15px;**

**}**

Аналогично сетке товаров выше, создаем стиль для похожей сетки:

**.feedbackgrid\_div {**

**border: 10px solid #D0F0F6;**

**margin-top: 5px;**

**display: grid;**

**background-color: rgba(208, 240, 246, .4);**

**padding: 30px;**

**border-radius: 40px;**

**grid-template-columns:;**

**gap: 16px;**

**padding: 16px;**

**}**

**.feedbackgrid\_div > div {**

**border-radius: 15px;**

**background-color: #82DCED;**

**border: 5px solid #D0F0F6;**

**padding: 3px;**

**margin: 5px;**

**font-size: 25px;**

**text-align: center;**

**}**

**.feedbackgridinside\_div {**

**background-color: rgba(208, 240, 246, .65);**

**padding: 2px;**

**border-radius: 7px;**

**}**

Добавляем стили для черной и оранжевой звезды отзыва:

**.star{**

**★**

**}**

**.starchecked {**

**color: orange;**

**}**

И в конце добавим стиль для нижнего копилефт блока страницы:

**.copyleft\_div {**

**margin: 0px;**

**margin-top: 50px;**

**margin-bottom: 0px;**

**width: 100%;**

**background-color: #bf564e;**

**color: white;**

**text-align: left;**

**padding: 0px;**

**padding-bottom: 10px;**

**}**

**2.3.4 Создание mensclothing.html страницы**

…

**3. Результат**

Во время выполнения курсовой работы были созданы:

1. Три HTML страницы сайта спортивной одежды;
2. CSS стили для данных страниц;
3. JavaScript скрипты для динамического взаимодействия со страницей.

Также в результате создана web-страница, удовлетворяещая требованиям заказчика, удобная пользователей и соответствующая современным стандартам.

Ссылка на main.html страницу:

Ссылка на репозиторий страницы:

**4. Литература**

1. [https://developer.mozilla.org/ru-RU/docs/Learn\_web\_development/Getting\_started/Web\_standards/How\_the\_web\_works](https://developer.mozilla.org/en-US/docs/Learn_web_development/Getting_started/Web_standards/How_the_web_works)
2. <https://www.w3schools.com/js/js_intro.asp> (перевод)
3. [https://developer.mozilla.org/](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started" \l "what_is_html)ru-RU/docs/Learn/HTML/Introduction\_to\_HTML/Getting\_started#what\_is\_html
4. <https://developer.mozilla.org/ru/docs/Learn_web_development/Core/Styling_basics/What_is_CSS>
5. <https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Grammar_and_types>