**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**

[**ВВЕДЕНИЕ 4**](#_Toc198585694)

[**1. ТЕОРИТИЧЕСКАЯ ЧАСТЬ 6**](#_Toc198585695)

[**1.1 Web-разработка 6**](#_Toc198585696)

[**1.2 HTML 7**](#_Toc198585697)

[**1.3 CSS 9**](#_Toc198585698)

[**1.4 JavaScript 11**](#_Toc198585699)

[**2. ПРАКТИЧЕСКАЯ ЧАСТЬ 13**](#_Toc198585700)

[**2.1 Цель курсовой работы и использованные технологии 13**](#_Toc198585701)

[**2.2 Среда создания и структура проекта 13**](#_Toc198585702)

[**2.3 Создание стилей, JS скриптов и верстка страниц 14**](#_Toc198585703)

[**2.3.1 Создание main.css 14**](#_Toc198585704)

[**2.3.2 Верстка main.html страницы 18**](#_Toc198585705)

[**2.3.3 Создание mensclothing.css 23**](#_Toc198585706)

[**2.3.4 Верстка mensclothing.html страницы 28**](#_Toc198585707)

[**2.3.5 Создание womensclothing.css 32**](#_Toc198585708)

[**2.3.6 Верстка womensclothing.html страницы 33**](#_Toc198585709)

[**2.3.7 Создание aboutus.css 34**](#_Toc198585710)

[**2.3.8 Верстка aboutus.html страницы 35**](#_Toc198585711)

[**2.3.9 Программирование changelang.js скрипта 36**](#_Toc198585712)

[**2.3.10 Программирование resizeSelect.js скрипта 38**](#_Toc198585713)

[**2.3.11 Программирование search.js скрипта 39**](#_Toc198585714)

[**3. РЕЗУЛЬТАТ 40**](#_Toc198585715)

[**4. ЛИТЕРАТУРА 40**](#_Toc198585716)

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 порядке):

1. **<a>** используется для создания гиперссылки на внешнюю страницу (сайт)
2. **<b>** делает текст жирным, без каких-либо других атрибутов
3. **<body>** содержит главное содержимое HTML-документа, например заголовки, абзацы, изображения, гиперссылки и т.д. В HTML-документе может быть только один <body>
4. **<br>** делает перенос строки (не требует </br>)
5. **<dd>** определяет описание в <dl> листе, часто используется под <dt> элементом
6. **<details>** указывает дополнительные сведения, которые пользователь может открывать и закрывать (спойлер)
7. **<dialog>** определяет диалоговое окно или подокно
8. **<div>** используется как контейнер для элементов HTML, которые затем стилизуются с помощью CSS или обрабатываются с помощью JavaScript
9. **<dl>** определяет список описаний. <dl> используется вместе с <dt> и <dd>
10. **<dt>** определяет имя в списке описаний. <dt> используется вместе с <dl> и <dd>
11. **<footer>** используется для выделения информации об авторстве, информации об авторских правах, контактную информацию и т.д.
12. **<h1>**, **<h2>**, ... , **<h6>** используются для определения заголовков. <h1> яаляется самым "важным" (большим) заголовком, а <h6> — наименее важным (мелким)
13. **<head>** является контейнером для метаданных и размещается между тегом <html> и тегом <body>
14. **<header>** — контейнер, который может содержать заголовки, значок, авторство
15. **<html>** представляет собой корень HTML-документа и является контейнером для всех остальных элементов HTML (кроме <!DOCTYPE>)
16. **<i>** выделяет часть текста курсивом
17. **<input>** определяет поле ввода, в которое пользователь может вводить данные
18. **<li>** определяет элемент списка. <li> используется внутри упорядоченных (<ol>) и неупорядоченных списков (<ul>), а также в списках меню (<menu>)
19. **<menu>** создает неупорядоченный список. <menu> является альтернативой тегу <ul>, и браузеры будут обрабатывать эти два списка одинаково.
20. **<ol>** определяет упорядоченный список. Может быть числовым или алфавитным. Используется совместно с <li> для определения элементов списка
21. **<option>** определяет опцию в выпадающем списке <select>, которую пользователь может выбрать
22. **<p>** определяет абзац. Браузеры автоматически добавляют одну пустую строку до и после каждого элемента <p>
23. **<s>** указывает текст, который больше не является правильным и/или актуальным. Текст будет отображаться зачеркнутым. <s> схож с <del>, но их не надо путать
24. **<section>** определяет раздел в документе. <section> может использоваться с <h1>...<h6> и <p>
25. **<select>** используется для создания выпадающего списка, используется вместе с <option> и часто задается в <form>. Позволяет пользователю выбрать одну из опций выпадающего списка
26. **<strong>** используется для определения текста высокой важности. Содержимое отображается жирным шрифтом. Не следует путать с <b>
27. **<style>** определяет информацию о стиле HTML документа с помощью CSS. Элемент <style> должен располагаться в разделе <head>
28. **<table>** тег определяет HTML-таблицу. HTML-таблица состоит из одного элемента <table> и одного или нескольких элементов <td>, <th> и <tr>
29. **<td>** определяет стандартную ячейку данных в HTML-таблице. Должен находиться внутри <tr>
30. **<th>** определяет ячейку заголовка в HTML-таблице. Должен находиться внутри <tr>
31. **<tr>** определяет строку в таблице HTML-таблице. Содержит один или несколько элементов <th> или <td>
32. **<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

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.

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.

Когда браузер отображает документ, он должен совместить его содержимое с его стилями. Этот процесс идёт в несколько этапов:

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»)

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 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 6 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**. "Hello World"
  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**.

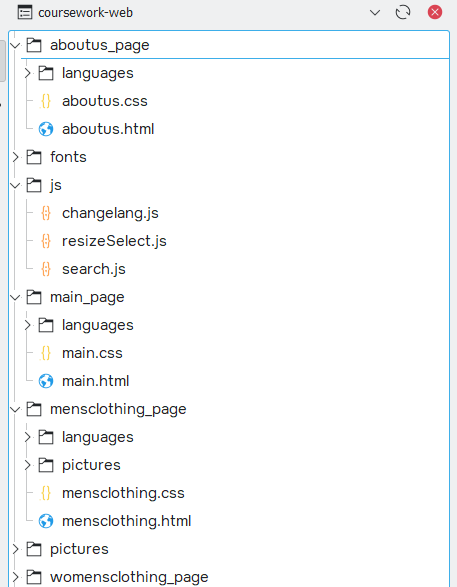
Все файлы и директории, используемые в создаваемом сайте хранятся в одной директории **coursework-web/** (корневая директория проекта).

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 2.1**.



**Рисунок 2.1. Структура проекта**

2.3 Создание стилей, JS скриптов и верстка страниц

### 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 {**

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

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

**}**

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;**

**}**

* **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%;**

**}**

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;**

**}**

**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**.

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 Верстка main.html страницы

В этой части будет описано пошаговое создание main.html страницы, т.е. главной страницы курсовой работы. Сначала добавим в main.html:

**<!DOCTYPE>**,

**<html lang="ru">**

**<head>**

**<meta charset="UTF-8">**

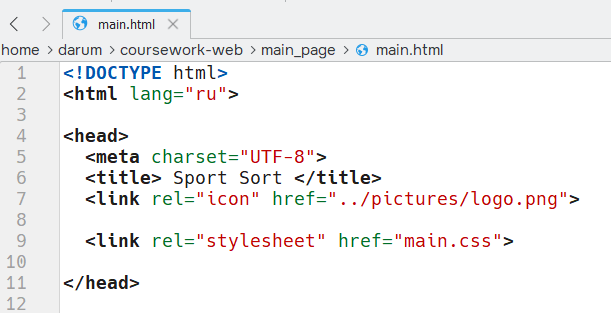
**<title> Sport Sort </title>**

**<link rel="icon" href="../pictures/logo.png">**

**<link rel="stylesheet" href="main.css">**

**</head>**

Тег <DOCTYPE> указывает что данный документ является документом HTML. <html lang="ru">, открывает главный тег HTML, в который войдет вся наша страница. <head> содержит метадату (заголовок страницы, стили и т.д.). <title> указывает заголовок нашей страницы. <link…> связывает внешние обьекты с нашей страницей – в данном случае он задает иконку страницы и добавляет стили CSS.



**Рисунок 2.2**

Далее мы создаем <body>, в котором будет основная структура нашей страницы. Добавим JS скрипты:

**<script src="../js/changelang.js"> </script>**

**<script src="../js/resizeSelect.js"> </script>**

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

Создаем специальный блок, который даствозможность распределения блоков в самом верху или низу страницыc атрибутом **class****="flex\_div"**:

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

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

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

**<img src="../pictures/logo.png" alt="LOGO" width="150">**

**<h1 style=****"font-size: 65px"> Sport Sort </h1>**

**<h2** **style="font-size: 30px">** **<p lang-attr=topbanner1> Дешево и сердито </p> </h2>**

**<div style="text-align: left">**

**<h2 lang-attr=langcur style="****margin: 0px">Язык</h2>**

**<select** **id="langSelect"** **onchange="changeLang(); resizeSelect(this)">**

**<option value="ru" selected>Русский</option>**

**<option value="en">English</option>**

**</select>** **</div>**  **</div>**

**Объяснение атрибутов:**

**style="font-size: 30px"** задает размер шрифта;

**lang-attr** — кастомный атрибут, используемый для перевода;

**style="text-align: left"** — меняет размещение текста;

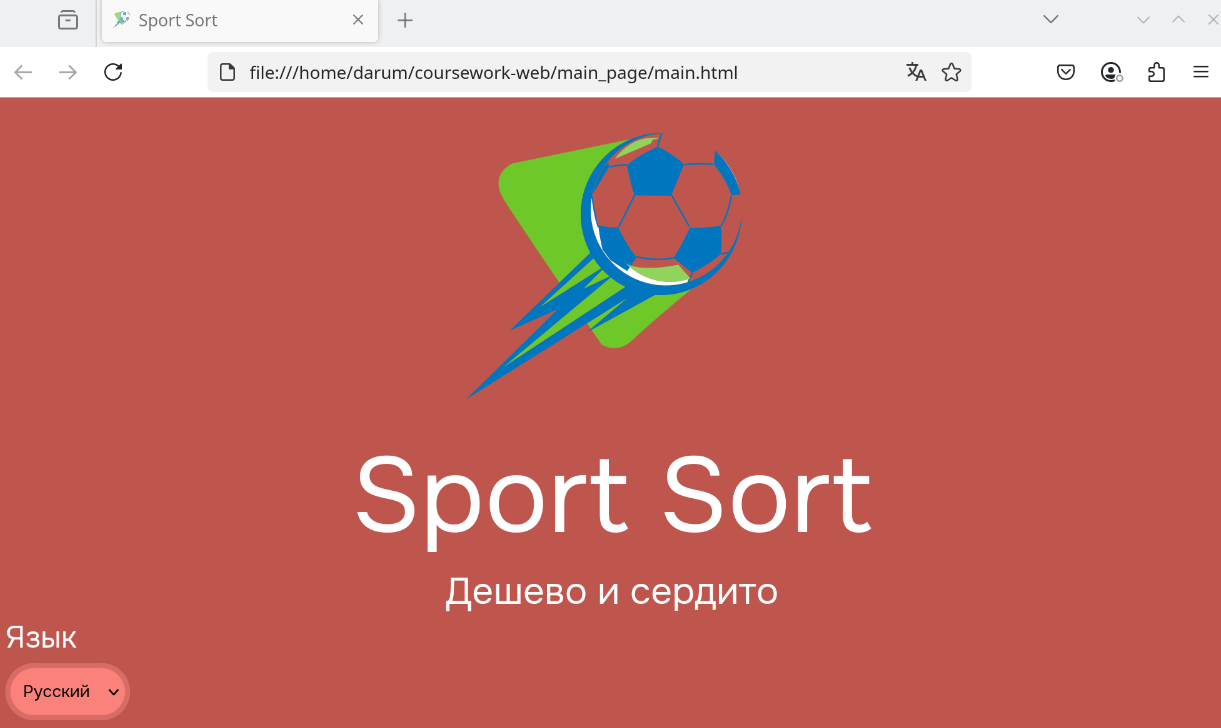
**margin: 0px** меняет отступ в пикселях со всех сторон объекта (текста);

**onchange="changeLang(); resizeSelect(this)"** выполняет две функции, при каждой смене выбранного варианта в select.

****

**Рисунок 2.3**

Полученный баннер можно рассмотреть на **рисунке 2.4**:

****

**Рисунок 2.4**

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

**<a** **href="../mensclothing\_page/mensclothing.html"** **style="text-decoration:none">**

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

**<h1> <b lang-attr=mensclothing>Мужской отдел</b> </h1>**

**<p><h3 lang-attr=mensportcl>Мужская спортивная одежда</h3></p>**

**</div>** **</a>**

Здесь тег <a> задает внешнюю ссылку через атрибут href.

Далее создаем такую же ссылку для страницы **womensclothing.html**.

**

**Рисунок 2.5**

Полученный результат *(рисунок 2.6)*:



**Рисунок 2.6**

Наконец, добавляем нижний баннер нашей страницы:

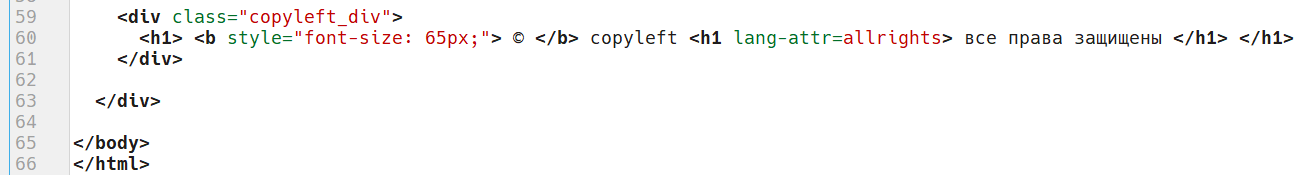
**<div class="copyleft\_div">**

**<h1> <b** **style="font-size: 65px"> 🄯 </b> copyleft <h1 lang-attr=allrights> все права защищены </h1> </h1>**

**</div>**

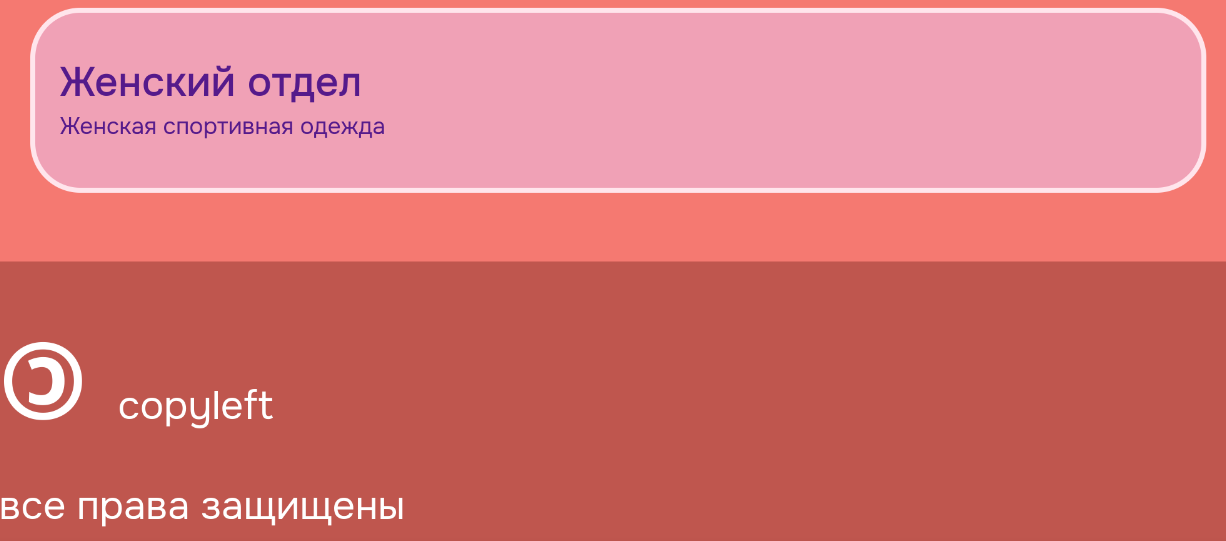
Под конец, закрываем <div class="flex\_div">, <body> и <html>:

**</div> </body> </html>**



**Рисунок 2.7**

Копилефт баннер можно рассмотреть на **рисунке 2.8**:

**Рисунок 2.8**

### 2.3.3 Создание mensclothing.css

Для начала создадим заранее **mensclothing.css** стиль для нашей будующей **mensclothing.html** страницы. Стили схожи с main.css, но с некоторыми отличиями:

Cтиль для элемента select, со скругленной полупрозрачной обводкой, отступами, цветами 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;**

**}**

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);**

**}**

Стиль для разделительного блока:

**.spacer\_div {**

**flex: 1;**

**}**

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

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;**

**}**

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;**

**}**

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;**

**}**

Стиль блока отзывов:

**.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;**

**}**

Аналогичные сетке товаров, стили для сетки отзывов:

**.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;**

**}**

Стили для черной и оранжевой звезды отзыва:

**.star{**

**★**

**}**

**.starchecked {**

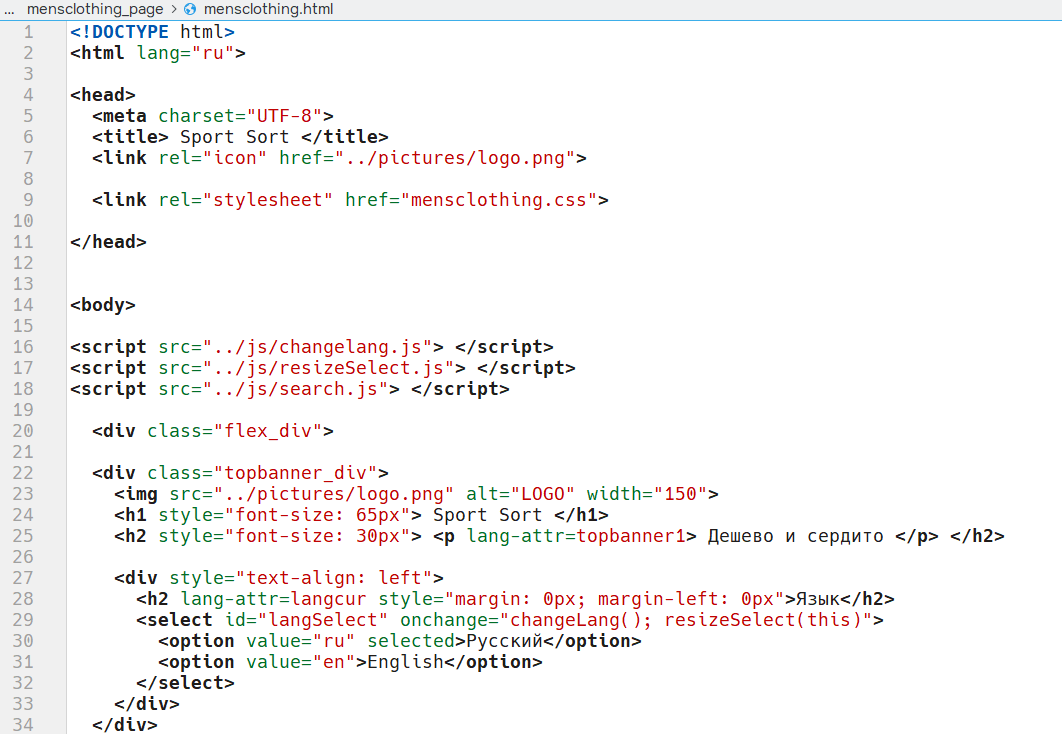
**color: orange;**

**}**

### 2.3.4 Верстка mensclothing.html страницы

Начинаем нашу страницу также, как main.html, с <html> тега и <body> с <div class="topbanner\_div">, за исключением добавления в <body>:

**<script src="../js/search.js"> </script>**. Этот скрипт реализации поиска опишем позже.



**Рисунок 2.9**

Далее создаем элемент поиска:

**<input type="text" id="search" placeholder="🔍" />**

Тег <input> позволяет пользователям вводить текстовые данные в строку.

Добавляем <div> с классом “grid\_div”, который будет динамической сеткой для наших товаров, и сами товары:

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

**<div class="item">**

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

**<img src="pictures/nikejacket.png" alt="Nike" width="300" height="300">**

**</div>**

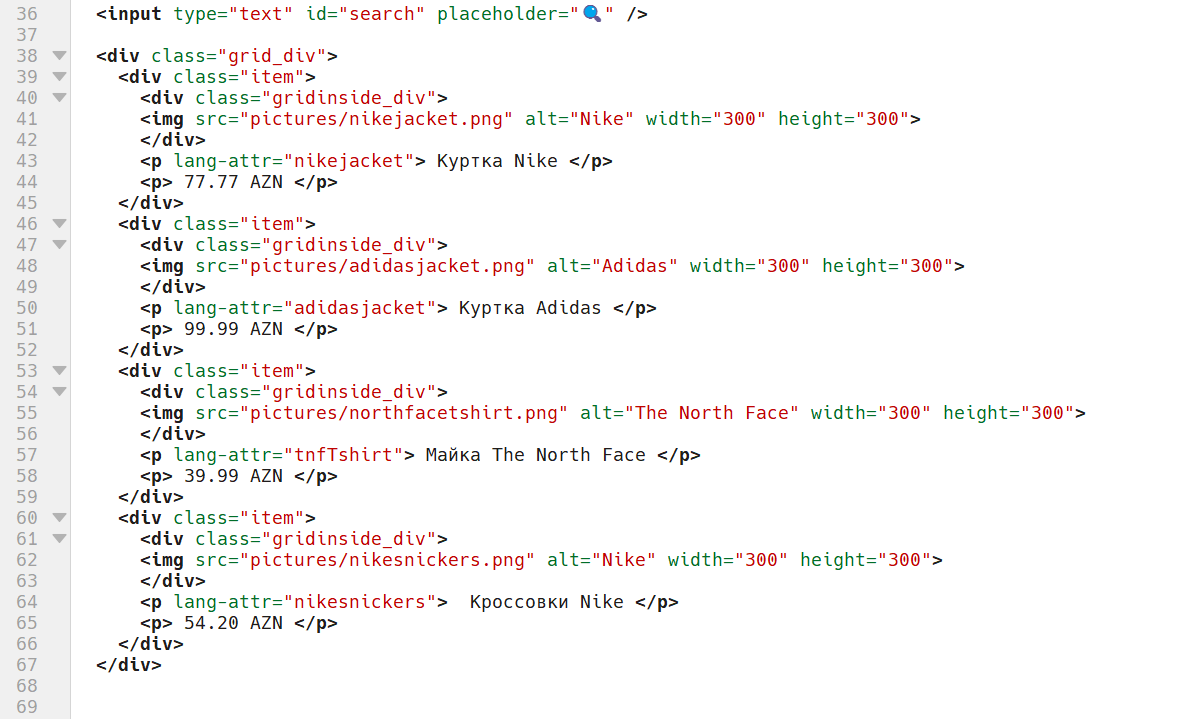
**<p lang-attr="nikejacket"> Куртка Nike </p>**

**<p> 77.77 AZN </p>**

**</div>**

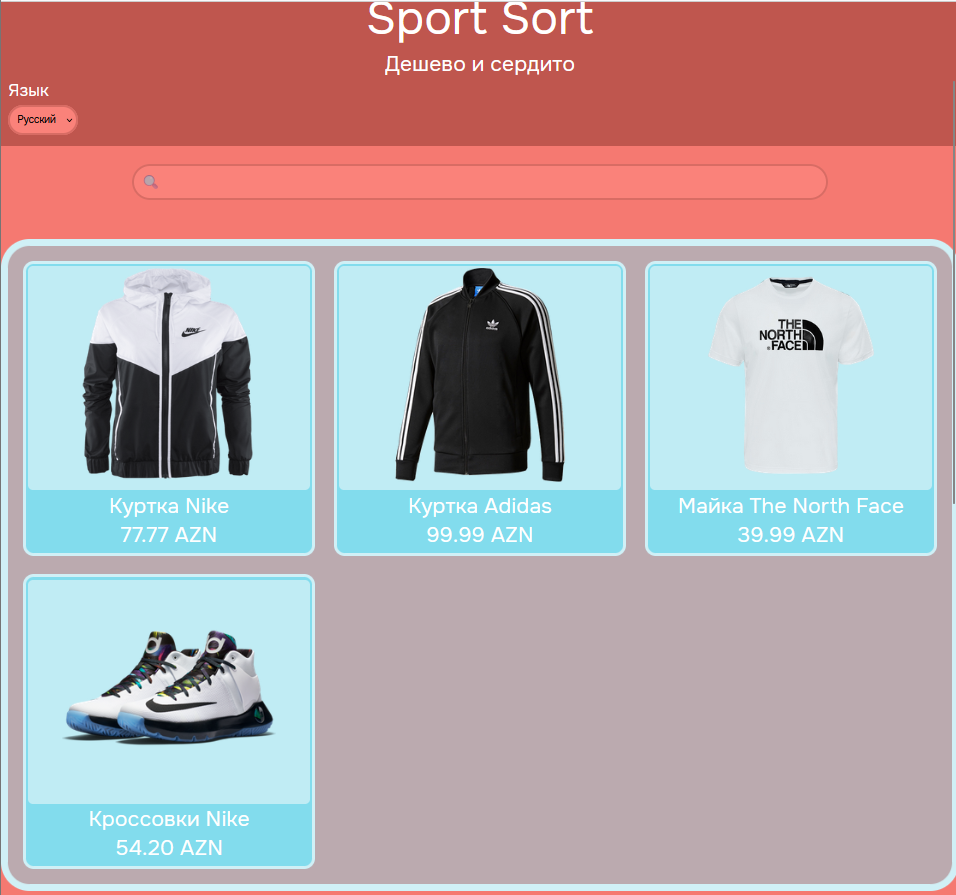
…

**</div>**

****

**Рисунок 2.10**

Полученный результат можно увидеть на рисунке 2.11:

****

**Рисунок 2.11**

Далее добавляем <div> c классом feedback\_div, который будет служить разделом для отзывов от пользователей:

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

**<h1> <b lang-attr="feedback"> Отзывы </b> </h1>**

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

**<div>**

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

**<h1> <b> John Doe </b> </h1>**

**<p lang-attr="nikejacket"> <b> Куртка Nike </b> </p>**

**</div>**

**<span class="starchecked">★</span>**

**<span class="starchecked">★</span>**

**<span class="starchecked">★</span>**

**<span class="starchecked">★</span>**

**<span class="star">★</span>**

**<p> Хорошое качество, доставили за 3 дня, рекомендую! </p>**

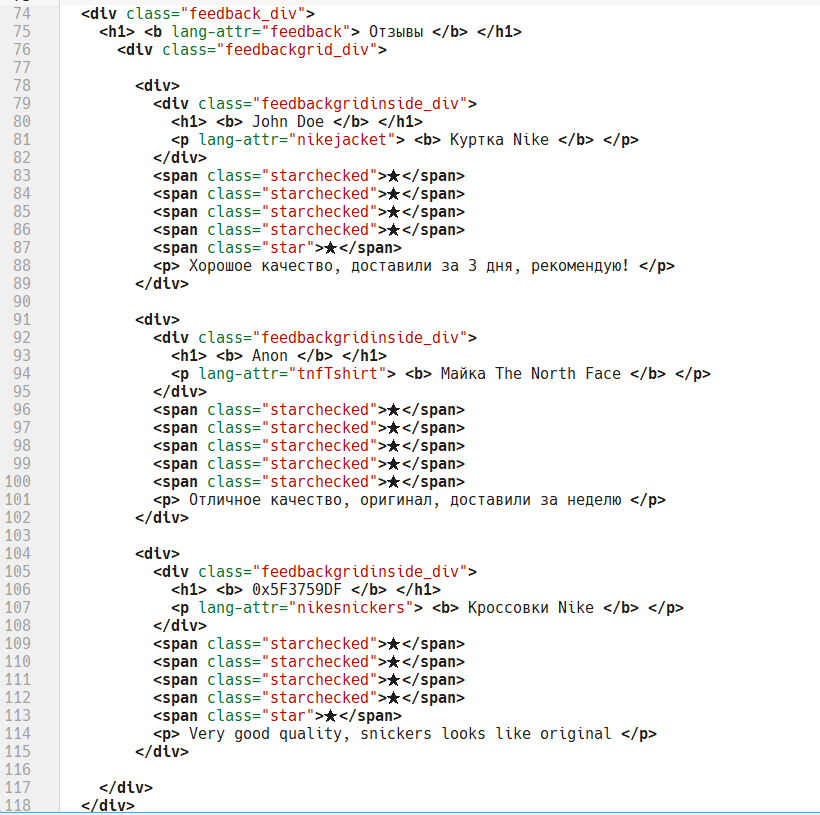
**</div>**

…

**</div>**

**</div>**

Класс starchecked, который мы задали в mensclothing.css, просто изменяет цвет звезд на оранжевый.



**Рисунок 2.12**

Результат можно увидеть на рисунке 2.13:

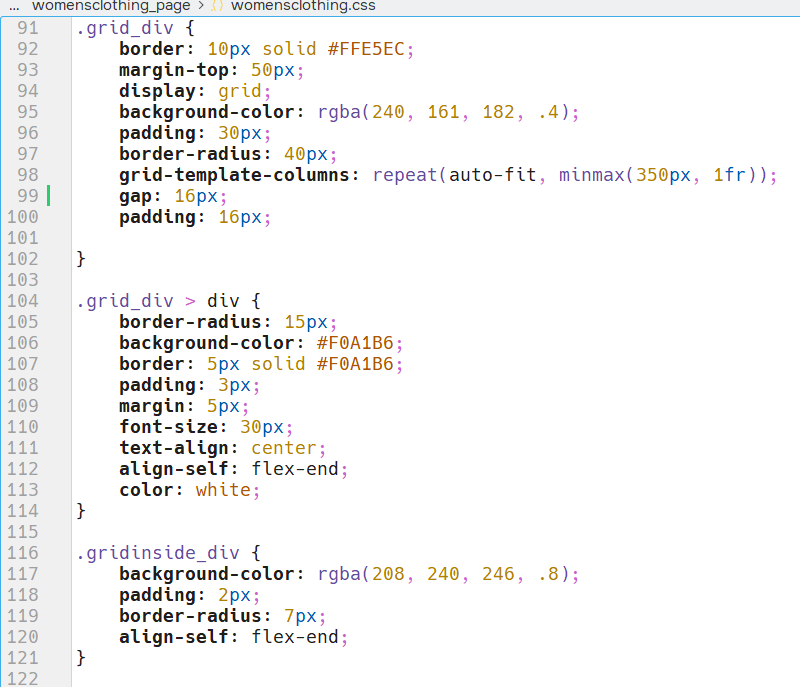
****

**Рисунок 2.13**

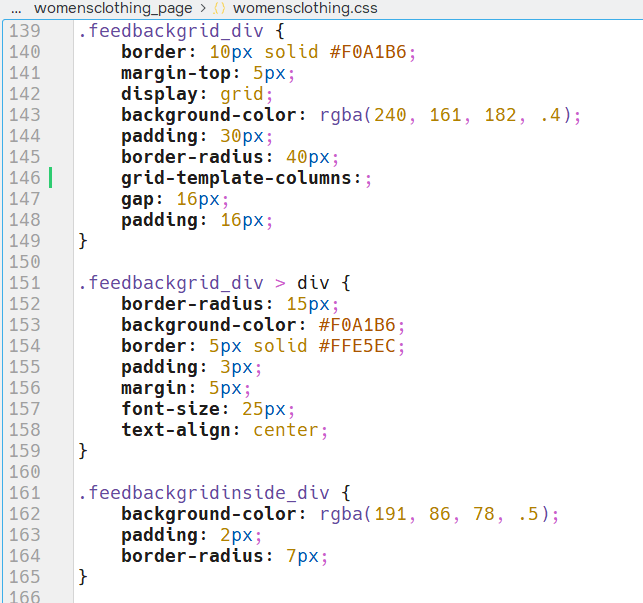
Под конец добавляем copyleft\_div <div> и закрываем <body> и <html>, также как в верстке main.html.

### 2.3.5 Создание womensclothing.css

Стиль страницы womensclothing.html почти идентична mansclothing.html, за исключением цветов страницы.



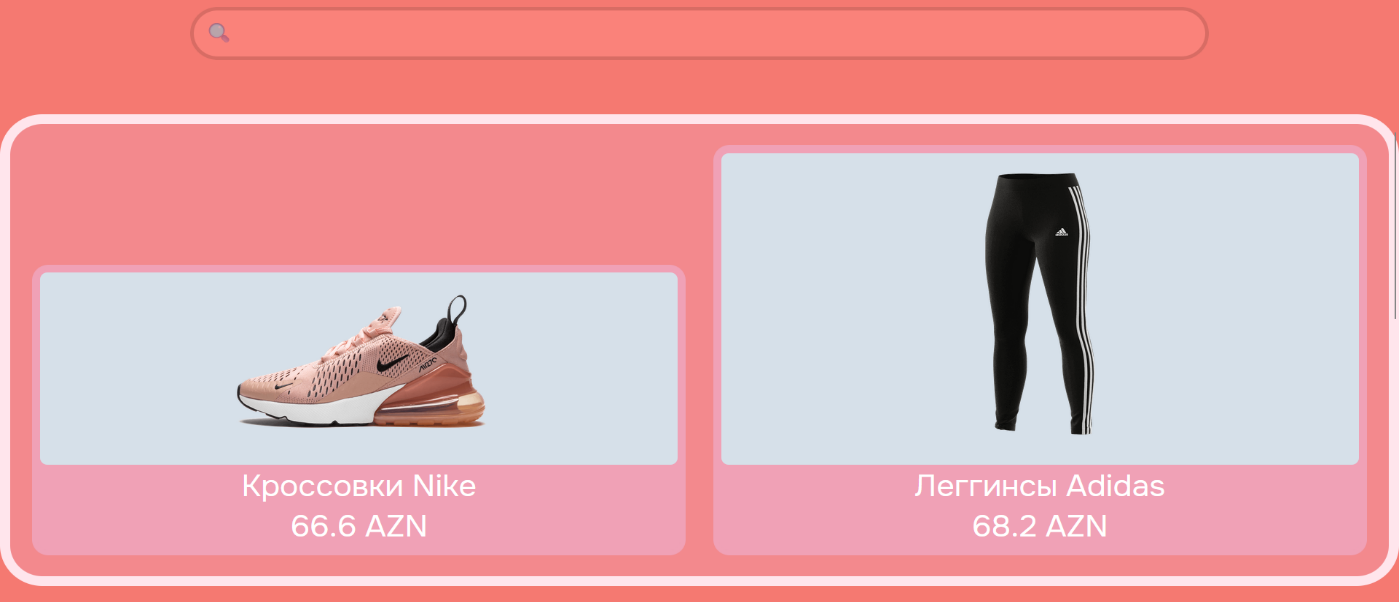
**Рисунок 2.14**

****

**Рисунок 2.15**

### 2.3.6 Верстка womensclothing.html страницы

Верстка womansclothing.html страницы не сильно отличается от верстки mansclothing.html страницы, отличие лишь в цветах, количестве товаров и их изображениях, а также в отзывах.



**Рисунок 2.16**

****

**Рисунок 2.17**

### 2.3.7 Создание aboutus.css

Далее мы создаем стили для страницы “О нас” (aboutus.html). После добавления стандартных стилей, взять из main.css (т.е. общий стиль html, body, .topbanner\_div и т.д), создаем стиль .aboutus\_div для специального <div>, который будет содержать информацию:

**.aboutus\_div {**

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

**border-radius: 40px;**

**margin-bottom: 5px;**

**margin-left: auto;**

**margin-right: auto;**

**width: 95%;**

**background-color: #ffffff;**

**text-align: left;**

**padding-left: 20px;**

**padding-right: 20px;**

**padding-top: 35px;**

**padding-bottom: 35px;**

**}**

### 2.3.8 Верстка aboutus.html страницы

После добавления стандартных элементов, взятых из предыдущих страниц, добавляем 2 баннера с информацией “О нас”:

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

**<h1><b lang-attr=aboutus>О нас</b></h1>**

**<p><h3 lang-attr=aboutusdesc>Добро пожаловать в Sport Sort! Мы - ваш надежный партнер в мире спортивной одежды. Наша миссия - предоставлять качественную, стильную и доступную одежду для активного образа жизни. Наша компания стремится поддерживать спортсменов и любителей фитнеса, предлагая широкий ассортимент продукции.</h3></p>**

**<p><h3 lang-attr=aboutusdesc2>Мы гордимся нашим вниманием к деталям, экологичным подходом к производству и отличным обслуживанием клиентов. Присоединяйтесь к нам и откройте для себя мир комфорта и стиля!</h3></p>**

**</div>**

**<div class="aboutus\_div" style="width: 50%; text-align: center;">**

**<h2 lang-attr="mail">Электронная почта</h2>**

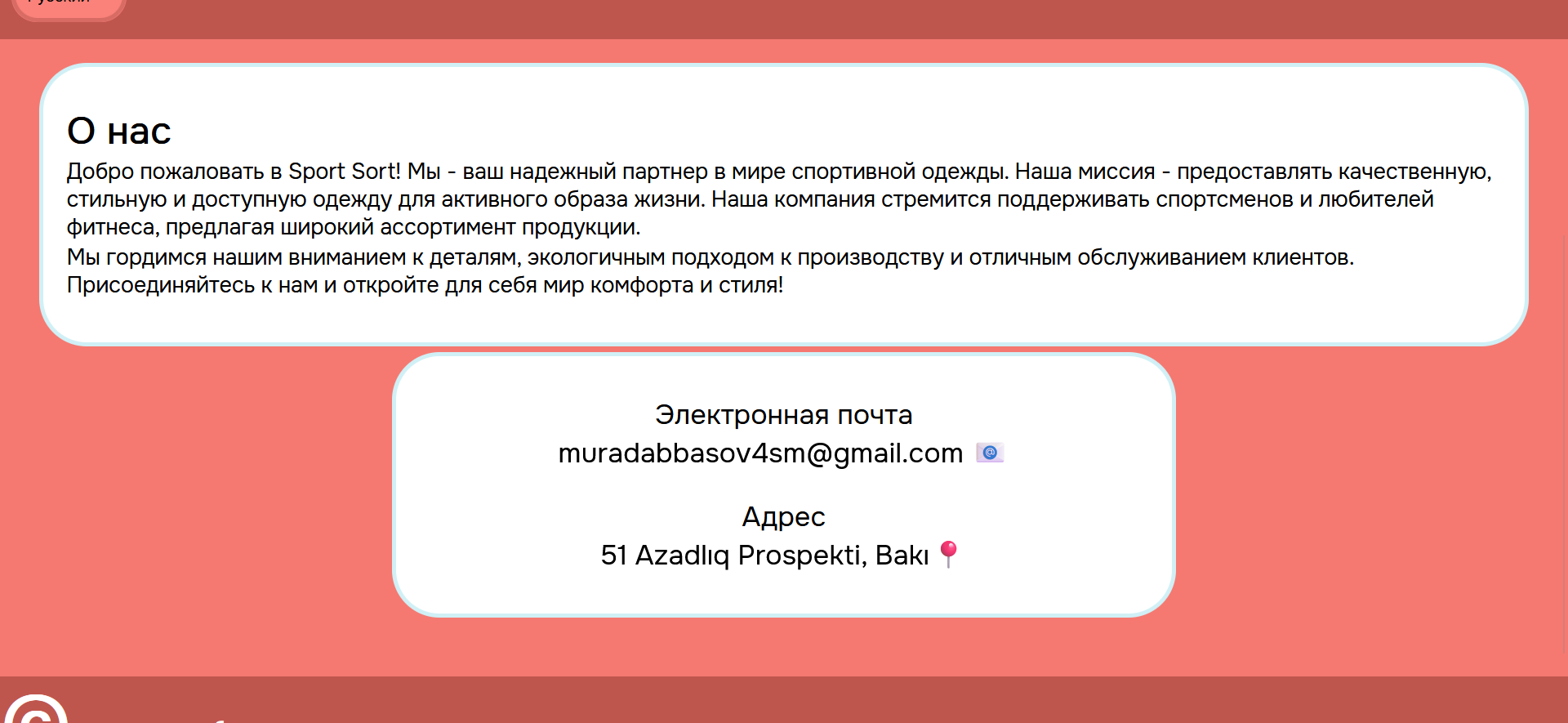
**<h2>muradabbasov4sm@gmail.com 📧</h2>**

**<p style="margin: 25px;"></p>**

**<h2 lang-attr="location">Адрес </h2>**

**<h2> 51 Azadlıq Prospekti, Bakı📍</h2>**

**</div>**

****

**Рисунок 2.18**

### 2.3.9 Программирование changelang.js скрипта

Данный скрипт предназначен для динамической смены языка страницы. Для начала создаем объект, служащий кэшем:

**const cache = {};**

Далее создаем асинхронную функцию для обновления языка:

**async function updateLang(lang) {**

Задаем элемент **document.cookie** создает куки-файлы, за счет которых происходит сохранение выбранного языка между страницами:

**document.cookie = `lang=${lang};path=/`;**

Устанавливаем атрибут **lang** для элемента **<html>**, указывая текущий язык документа:

**document.documentElement.lang = lang;**

Добавляем в **<html>** атрибут **data-timestamp** с текущей временной меткой для отслеживания изменений:

**document.documentElement.dataset.timestamp = Date.now();**

Ищем элемент **<select>** с **id="langSelect"** для выбора языка (элемент должен быть один на всю страницу), если он существует меняем значение:

**const select = document.getElementById('langSelect');**

**if (select) select.value = lang;**

Проверяем, если язык **ru**, то используем пустой объект переводов. Скрипт пытается взять переводы из **cache[lang]**, иначе загружает их асинхронно:

**const translations = lang === 'ru' ? {} : cache[lang] || (await new Promise(r => {**

Создаем элемент s и задаем ему путь к нашему файлу с переводами:

**const s = document.createElement('script');**

**s.src = `./languages/${lang}.js`;**

Загружаем скрипт, содержащий перевод — при успехе мы переносим перевод из translation\_${lang} в наш cache, при неудаче возвращаем пустоту:

**s.onload = () => r(cache[lang] = window[`translation\_${lang}`] || {});**

**s.onerror = () => r({});**

Добавляем созданный **<script>** в **<head>** документа, инициируя загрузку файла, а также закрываем конструкцию **Promise**:

**document.head.appendChild(s);**

**}));**

Далее перебираем все элементы с атрибутом **lang-attr** и сохраняем исходный текст из них в **el.dataset.ru**:

**document.querySelectorAll('[lang-attr]').forEach(el => {**

**const key = el.getAttribute('lang-attr');**

**el.dataset.ru = el.dataset.ru || el.textContent;**

Проверяем, если язык **ru**, то мы просто загружаем обратно текст из **el.dataset.ru**, иначе берем перевод из **translations**:

**el.textContent = lang === 'ru' ? el.dataset.ru : translations[key] || el.dataset.ru;**

**});**

Изменяем все **<a>** теги на странице для сохранения языка при переходе и закрываем функцию:

**document.querySelectorAll('a[href]').forEach(a => {**

**const url = new URL(a.href, location.href);**

**url.searchParams.set('lang', lang);**

**a.href = url.toString();**

**});**

**}**

Создаем еще одну функцию для инициализации языка при запуске:

**function initLang() {**

Определяем текущий язык: сначала проверяем куки-файлы, затем параметр **lang** в **URL**, иначе используем **ru**:

**const lang = (`; ${document.cookie}`.split('; lang=')[1]?.split(';')[0]) ||**

**new URLSearchParams(location.search).get('lang') || 'ru';**

Находим элемент **<select>** с **id="langSelect"**, если **<select>** существует, добавляем обработчик события **onchange**, который вызывает **updateLang** в следующем кадре анимации и закрываем функцию:

**const select = document.getElementById('langSelect');**

**if (select) select.onchange = () => updateLang(select.value);**

**requestAnimationFrame(() => updateLang(lang));**

**}**

Добавляем обработчик события **DOMContentLoaded**, вызывающий **initLang** после загрузки **DOM**:

**addEventListener('DOMContentLoaded', initLang);**

Заканчиваем скрипт установив **initLang** как обработчик события **pageshow**, чтобы обновлять язык при возврате к странице при навигации назад в браузере:

**onpageshow = initLang;**

### 2.3.10 Программирование resizeSelect.js скрипта

Создаем **resizeSelect** функцию, которая будет динамически изменять размер элемента <select> в зависимости от длины текста:

**function** **resizeSelect(select) {**

Создаём элемент **<span>** в памяти и сохраняем его в **tmp** — он будет использоваться для измерения ширины текста:

**const tmp = document.createElement("span");**

Задаем стили **visibility=hidden** (скрытый) и **position=absolute** (так элемент не будет влиять на расположение остальных элементов) для элемента **<span>** в **tmp**, а также шрифт, использованный в **<select>**:

**tmp.style.visibility = "hidden";**

**tmp.style.position = "absolute";**

**tmp.style.font = getComputedStyle(select).font;**

Приравниваем текст внутри элемента **tmp** к тексту в текущем **<option>** в элементе **<select>**:

**tmp.innerText = select.options[select.selectedIndex].text;**

Добавляем элемент **tmp** в **<body>**, приравниваем ширину элемента **<select>** к **(tmp.offsetWidth + 50)** пикселей, удаляем элемент **tmp** и закрываем **resizeSelect** функцию:

**document.body.appendChild(tmp);**

**select.style.width = (tmp.offsetWidth + 50) + "px";**

**document.body.removeChild(tmp);**

**}**

Под конец, вызываем нашу **resizeSelect** функцию сначала для элементов с идентификатором **langSelect**, потом **dynamicSelect**:

**resizeSelect(document.getElementById("langSelect"));**

**resizeSelect(document.getElementById("dynamicSelect"));**

### 2.3.11 Программирование search.js скрипта

Последним опишем скрипт для реализации поиска элементов.

Добавим обработчик события **DOMContentLoaded**, который срабатывает когда HTML-документ полностью загружен (только сам документ):

**addEventListener('DOMContentLoaded', () => {**

Создаем переменную **search**, которая ссылается на элемент с идентификатором **search**:

**const search = document.getElementById('search');**

Проверяя наличие переменной, добавляем еще одно событие — **input**, который срабатывает каждый раз когда пользователь делает ввод:

**if (search) {**

**search.addEventListener('input', () => {**

Создаем переменную **query**, которая содержит наш ввод из **search**, переведенный в нижний регистр (так наш поиск будет не чувствительным к регистру):

**const query = search.value.toLowerCase();**

Находим все элементы класса **item**, переводим их в нижний регистр и выполняем для каждого из них следующее — если текст элемента содержит текст из **query**, то задается **display: block** (видимый объект), в ином случае **display: none** (скрытый объект):

**document.querySelectorAll('.item').forEach(item => {**

**item.style.display = item.textContent.toLowerCase().includes(query) ? 'block' : 'none';**

Заканчиваем скрипт, закрывая цикл **forEach**, обработчик **input**, условие **if (search)** и сам обработчик **DOMContentLoaded**:

**});**

**});**

**}**

**});**

3. Результат

Во время выполнения курсовой работы были созданы:

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

В результате создана web-страница, удовлетворяещая требованиям заказчика, удобная пользователям 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#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>