**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

# **Дисциплина:** Фронт-энд разработка

Отчет

### Лабораторная работа 1: вёрстка сайта средствами HTML, CSS и Bootstrap

Выполнила: Пронина Александра

Группа К33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

Задача: Выполнить верстку для веб-приложения (средствами HTML,CSS, bootstrap), которое помогает путешественникам планировать и организовывать свои путешествия. Оно предоставляет пользователю возможность создавать маршруты, визуально отображать их на карте и управлять всей информацией о поездках в одном месте.

Итог: Мной было создано 5 базовых страниц с возможностью расширения функционала. Их код представлен и пояснен ниже.

1. Страница для входа (Entry page):

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Вход</title>  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>  
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>  
 <style>  
 */\* Custom CSS \*/* .container {  
 margin-top: 50px;  
 max-width: 400px;  
 position: relative;  
 }  
 .fog {  
 content: "";  
 position: absolute;  
 top: 0;  
 left: 0;  
 height: 100%;  
 width: 100%;  
 background-size: cover;  
 overflow: hidden;  
 background-repeat: repeat-x;  
 background-image: url("../css/img/fog-2.png");  
 background: url("../css/img/background.png") center center;  
  
 background-size: cover;  
 z-index: -1;  
 animation: fog 20s;  
 }  
 @keyframes fog {  
 0% {  
 opacity: 0.05;  
 }  
  
 100% {  
 opacity: 1;  
 }  
  
 }  
  
 .form {  
 position: relative;  
 z-index: 1;  
 }  
 .halloween-btn {  
 background-color: #ff7518;  
 color: #ffffff;  
 border: none;  
 box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.5);  
 animation: fog 20s;  
 }  
 .halloween-btn:hover {  
 box-shadow: none;  
 }  
 */\* Custom CSS for neon letters \*/* h2 {  
 color: #fff3cd; */\* Зеленый цвет \*/* text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;  
 animation: fog 20s;  
 }  
 .form-control {  
 color: #ff7518; */\* Зеленый цвет \*/* background-color: #ffffff; */\* Черный фон \*/* border-color: #ff7518; */\* Зеленая граница \*/* box-shadow: 0px 0px 10px #ff7518;  
 animation: fog 20s;  
  
 }  
 body {  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 min-height: 100vh;  
  
 }  
  
 </style>  
</head>  
  
<body>  
  
<div class="container">  
 <h2>Вход</h2>  
 <form>  
 <div class="form-group">  
 <input type="email" class="form-control" id="email" placeholder="Введите email" required>  
 </div>  
 <div class="form-group">  
  
 <input type="password" class="form-control" id="password" placeholder="Введите пароль" required>  
 </div>  
 <button type="submit" class="halloween-btn">Войти</button>  
 </form>  
  
</div>  
<div class="fog"></div>  
  
<script>  
 *// JavaScript-код для валидации формы входа* $(document).ready(function(){  
 $('form').submit(function(e){  
 e.preventDefault();  
 var email = $('#email').val();  
 var password = $('#password').val();  
 *// Проверка правильности заполнения полей формы* if(email === '' || password === ''){  
 *alert*('Пожалуйста, заполните все поля');  
 } else {  
 *// Отправка данных на сервер для проверки  
 // и выполнение дальнейших действий* }  
 });  
 });  
</script>  
  
</body>  
</html>

1. В данном HTML-коде создается страница для входа в аккаунт. Она содержит форму входа с полями для ввода e-mail и пароля.

2. В <style> задаются пользовательские стили для страницы, такие как цвет фона, позиционирование элементов и оформление кнопки и текста.

3. Внутри <body> находится контейнер с классом container, в котором размещена форма входа.

4. В форме есть два поля ввода: email и пароль, стилизованные с помощью класса form-control.

5. Также присутствует кнопка "Войти" с классом halloween-btn, которая имеет свой собственный стиль. 6. Ниже формы находится <div> с классом fog, который создает эффект тумана на фоне страницы.

7. В <script> находится JavaScript-код для валидации формы входа. (используется jQuery

для обработки события отправки формы.)

8. При отправке формы проверяются значения полей email и пароля. Если одно из полей не заполнено, выводится сообщение об ошибке. Если оба поля заполнены, можно выполнить отправку данных на сервер для проверки и выполнения дополнительных действий.

1. Страница регистрации (Registration-page):

Этот код - HTML-страница, которая содержит форму регистрации.

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Регистрация</title>  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>  
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>  
 <style>  
 */\* Custom CSS \*/* .container {  
 margin-top: 50px;  
 max-width: 400px;  
 }  
 .background-img {  
 background: url("../css/img/background.png") center center;  
 background-size: cover;  
 width: 100%;  
 height: 100%;  
 z-index: -1;  
 content: "";  
 position: absolute;  
 top: 0;  
 left: 0;  
 overflow: hidden;  
 }  
 .form-control {  
 color: #ff7518; */\* Зеленый цвет \*/* background-color: #ffffff; */\* Черный фон \*/* border-color: #ff7518; */\* Зеленая граница \*/* box-shadow: 0px 0px 10px #ff7518;  
 animation: fog 20s;  
  
 }  
 h2 {  
 color: #fff3cd; */\* Зеленый цвет \*/* text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;  
 animation: fog 20s;  
 }  
 .halloween-btn {  
 background-color: #ff7518;  
 color: #ffffff;  
 border: none;  
 box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.5);  
 animation: fog 20s ;  
 }  
 .halloween-btn:hover {  
 box-shadow: none;  
 }  
 </style>  
</head>  
<body>  
<div class="background-img">  
 <div class="container">  
 <h2>Регистрация</h2>  
 <form>  
 <div class="form-group">  
 <label for="email">Email адрес:</label>  
 <input type="email" class="form-control" id="email" placeholder="Введите email" required>  
 </div>  
 <div class="form-group">  
 <label for="password">Пароль:</label>  
 <input type="password" class="form-control" id="password" placeholder="Введите пароль" required>  
 </div>  
 <div class="form-group">  
 <label for="confirm\_password">Подтвердите пароль:</label>  
 <input type="password" class="form-control" id="confirm\_password" placeholder="Подтвердите пароль" required>  
 </div>  
 <button type="submit" class="halloween-btn">Зарегистрироваться</button>  
 </form>  
</div>  
<script>  
 *// JavaScript-код для валидации формы регистрации* $(document).ready(function(){  
 $('form').submit(function(e){  
 e.preventDefault();  
 var email = $('#email').val();  
 var password = $('#password').val();  
 var confirm\_password = $('#confirm\_password').val();  
 *// Проверка правильности заполнения полей формы* if(email === '' || password === '' || confirm\_password === ''){  
 *alert*('Пожалуйста, заполните все поля');  
 } else if(password !== confirm\_password){  
 *alert*('Пароли не совпадают');  
 } else {  
 *// Отправка данных на сервер для регистрации пользователя  
 // и выполнение дальнейших действий* }  
 });  
 });  
</script>  
</div>  
</body>  
</html>

1. В начале кода устанавливается DTD (Document Type Definition), который указывает на тип документа, и указывается язык страницы (ru).

2. Метатеги <head> содержат информацию о странице, такую как charset (кодировка символов), viewport (настройка масштабирования), и title (заголовок страницы).

3. Далее идет подключение стилей Bootstrap и скриптов jQuery и Popper.js, которые помогут упростить разработку и сделать страницу адаптивной.

4. В <style> указывается пользовательский CSS для страницы. Здесь устанавливаются стили для контейнера, фонового изображения, полей ввода и кнопки.

5. В <body> размещена основная структура страницы. Внутри <div class="background-img"> находится контейнер с классом container, который содержит заголовок <h2> и форму регистрации <form>.

6. Внутри формы находятся поля ввода для email, пароля и подтверждения пароля. Используется класс form-control для стилизации полей ввода.

7. Также присутствует кнопка "Зарегистрироваться" с классом halloween-btn, которая имеет свой собственный стиль.

8. Внутри тега <script> находится JavaScript-код, который обрабатывает отправку формы регистрации. Он использует jQuery и следующие задачи:

- Предотвращает отправку формы по умолчанию (чтобы страница не перезагружалась).

- Получает значения полей ввода email, пароля и подтверждения пароля.

- Проверяет, заполнены ли все поля.

- Проверяет, соответствуют ли пароль и его подтверждение.   
- Если все проверки прошли успешно, можно отправить данные на сервер для регистрации пользователя и выполнять дополнительные действия.

Отдельное путешествие(the way page):

<!DOCTYPE html>  
<html lang="ru">  
  
<head>  
 <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800;900&display=swap" rel="stylesheet">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <title>Маршрут путешествия</title>  
 <style>  
 #map {  
 width: 50%;  
 height: 400px;  
 margin-top: 40px;  
 margin-left:50px ;  
  
 z-index: 1;  
 }  
 .navigation {  
  
 display: flex;  
 margin-top: 20px;  
 flex-direction: column;  
 align-items: flex-start;  
 margin-right: 150px;  
 float: right;  
 gap: 30px;  
 }  
 .navigation a {  
  
 font-size: 1.25em;  
 color: #fff;  
 font-weight: 500;  
 */\*margin-left: 75px;  
 padding: 5px 25px;\*/* transition: .5s;  
 }  
 .navigation a:hover {  
 color: #fff3cd;  
 text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;  
 }  
 .navigation a span {  
  
 top: 0;  
 left: 200px;  
 width: 100%;  
 height: 100%;  
 background: #ffffff fixed;  
 border-radius: 25px;  
 z-index: -1;  
 transform: scale(0);  
 opacity: 0;  
 transition: .5s;  
 }  
 .navigation a:hover span {  
 transform: scale(1);  
 opacity: 0.25;  
 }  
 h1 {  
 margin-top: 40px;margin-left: 50px;  
 font-size: 2em;  
 color: #fff3cd; */\* Зеленый цвет \*/* text-shadow: 0px 0px 10px #fe3901, 0px 0px 20px #ff7518, 0px 0px 30px #ffa801;  
  
  
 }  
 p{  
 color: #fff3cd;  
 margin-top: 90px;  
 margin-left: 750px;  
 }  
  
  
 .background-img {  
 background: url("../css/img/358.png") center;  
 background-size: cover;  
 width: 100%;  
 height: 100%;  
 z-index: -3;  
 content: "";  
 position: fixed;  
 top: 0;  
 left: 0;  
 overflow: hidden;  
 }  
 </style>  
</head>  
  
<body>  
<div class="background-img">  
 <h1>Маршрут путешествия по городу</h1>  
 <nav class="navigation">  
 <a href="#">Мои воспоминания<span></span></a>  
 <a href="#">Мои маршруты<span></span></a>  
 <a href="#">Проживание и питание<span></span></a>  
 </nav>  
 <div id="map"></div>  
</div>  
<script>  
 function *initMap*() {  
 *// Создаем карту* var map = new google.maps.Map(document.getElementById('map'), {  
 center: {  
 lat: 55.7558,  
 lng: 37.6176  
 },  
 zoom: 12  
 });  
  
 *// Указываем точки маршрута* var locations = [  
 {lat: 55.7558, lng: 37.6176},  
 {lat: 55.7522, lng: 37.6156},  
 {lat: 55.7517, lng: 37.6109},  
 {lat: 55.7495, lng: 37.6051},  
 {lat: 55.7539, lng: 37.6226}  
 ];  
  
  
 *// Создаем маркеры для каждой точки маршрута* locations.forEach(function (location) {  
 new google.maps.Marker({  
 position: location,  
 map: map  
 });  
 });  
  
 *// Создаем полилинию, соединяющую точки маршрута* var path = new google.maps.Polyline({  
 path: locations,  
 geodesic: true,  
 strokeColor: '#ff0000',  
 strokeOpacity: 1.0,  
 strokeWeight: 2  
 });  
  
 *// Включаем отображение полилинии на карте* path.setMap(map);  
 }  
</script>  
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyB41DRUbKWJHPxaFjMAwdrzWzbVKartNGg&callback=initMap" async defer></script>  
</body>  
  
</html>

Страница с поиском и фильтрацией(the searching and filter page):

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800;900&display=swap" rel="stylesheet">  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Мои путешествия</title>  
 <style>  
 .background-img {  
 background: url("../css/img/358.png") center;  
 background-size: cover;  
 width: 100%;  
 height: 100%;  
 z-index: -1;  
 content: "";  
 position: fixed;  
 top: 0;  
 left: 0;  
 overflow: hidden;  
 }  
 .filters {  
 margin-bottom: 20px;  
 }  
 .travel-item {  
 margin-bottom: 10px;  
 padding: 10px;  
 border: 1px solid #ccc;  
 border-radius: 5px;  
 }  
 .upcoming {  
 background-color: #cb6526;  
 }  
 .ongoing {  
 background-color: #ffa801;  
 }  
 .completed {  
 background-color: #981800;  
 }  
 .postponed {  
 background-color: #fed700;  
 }  
 .cancelled{  
 background-color: #fe3901;  
 }  
 .form-check{  
 color: #ffa801;  
 }  
 h2 {  
 margin-top: 30px;  
 color: #fff3cd; */\* Зеленый цвет \*/* text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;  
  
 }  
*/\*  
 h4 {  
 margin-top: 20px;  
 color: #fff3cd; /\* Зеленый цвет \*/  
 /\* text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;} \*/* .form-check {  
 display: inline-block;  
 margin-right: 100px;  
  
 }  
  
  
 body {  
 */\* Установите margin и padding на 0 для body, чтобы избежать сдвига и прокрутки страницы \*/* margin: 0;  
 padding: 0;  
 }  
 .scroll-container {  
 */\* Установите высоту на 100vh для контейнера прокрутки, чтобы заполнить весь экран \*/* height: 100vh;  
 overflow: scroll;  
 }  
 */\* Добавьте стили скроллбара \*/* .scroll-container::-webkit-scrollbar {  
 width: 10px;  
 }  
 .scroll-container::-webkit-scrollbar-track {  
 background-color: #f1f1f1;  
 }  
 .scroll-container::-webkit-scrollbar-thumb {  
 background-color: #888;  
 }  
 </style>  
</head>  
<body>  
<div class="background-img">  
 <div class="container">  
 <h2>Мои путешествия</h2>  
 <div class="filters">  
 <div class="form-check">  
 <input class="form-check-input" type="checkbox" id="upcomingFilter">  
 <label class="form-check-label" for="upcomingFilter">Предстоящие</label>  
 </div>  
 <div class="form-check">  
 <input class="form-check-input" type="checkbox" id="ongoingFilter">  
 <label class="form-check-label" for="ongoingFilter">В процессе</label>  
 </div>  
 <div class="form-check">  
 <input class="form-check-input" type="checkbox" id="completedFilter">  
 <label class="form-check-label" for="completedFilter">Завершенные</label>  
 </div>  
 <div class="form-check">  
 <input class="form-check-input" type="checkbox" id="postponedFilter">  
 <label class="form-check-label" for="postponedFilter">Отложенные</label>  
 </div>  
 <div class="form-check">  
 <input class="form-check-input" type="checkbox" id="cancelledFilter">  
 <label class="form-check-label" for="cancelledFilter">Отмененные</label>  
 </div>  
 </div>  
 <div class="scroll-container">  
 <div id="travelList">  
 *<!-- Здесь будут отображены путешествия -->* </div>  
 </div>  
 </div>  
</div>  
<script>  
 const travels = [  
 {  
 name: "Париж, Франция",  
 status: "upcoming",  
 date: "10.10.2022"  
 },  
 {  
 name: "Лондон, Великобритания",  
 status: "completed",  
 date: "01.01.2021"  
 },  
 {  
 name: "Токио, Япония",  
 status: "ongoing",  
 date: "15.05.2023",  
  
 },  
 {  
 name: "Рим, Италия",  
 status: "cancelled",  
 date: "20.07.2022"  
 }  
 ];  
  
 *// Обработчик события изменения фильтров* function *handleFilterChange*() {  
 const upcomingFilter = document.getElementById("upcomingFilter").checked;  
 const ongoingFilter = document.getElementById("ongoingFilter").checked;  
 const completedFilter = document.getElementById("completedFilter").checked;  
 const postponedFilter = document.getElementById("postponedFilter").checked;  
 const cancelledFilter = document.getElementById("cancelledFilter").checked;  
  
 const filteredTravels = travels.filter(travel => {  
 return (upcomingFilter && travel.status === "upcoming") ||  
 (ongoingFilter && travel.status === "ongoing") ||  
 (completedFilter && travel.status === "completed") ||  
 (postponedFilter && travel.status === "postponed")||  
 (cancelledFilter && travel.status === "cancelled");  
  
 });  
  
 *renderTravelList*(filteredTravels);  
 }  
  
 *// Отрисовка списка путешествий* function *renderTravelList*(travels) {  
 const travelListElement = document.getElementById("travelList");  
 travelListElement.innerHTML = "";  
  
 travels.forEach(travel => {  
 const travelItem = document.createElement("div");  
 travelItem.classList.add("travel-item");  
  
 switch (travel.status) {  
 case "upcoming":  
 travelItem.classList.add("upcoming");  
 break;  
 case "ongoing":  
 travelItem.classList.add("ongoing");  
 break;  
 case "completed":  
 travelItem.classList.add("completed");  
 break;  
 case "postponed":  
 travelItem.classList.add("postponed");  
 break;  
 case "cancelled":  
 travelItem.classList.add("cancelled");  
 break;  
  
 }  
  
 const travelName = document.createElement("h5");  
 travelName.textContent = travel.name;  
  
 const travelStatus = document.createElement("p");  
 travelStatus.textContent =travel.status;  
  
 const travelDate = document.createElement("p");  
 travelDate.textContent = travel.date;  
  
 const imgTr = document.createElement("img");  
 imgTr.textContent = travel.img;  
  
 travelItem.appendChild(travelName);  
 travelItem.appendChild(travelStatus);  
 travelItem.appendChild(travelDate);  
 travelItem.appendChild(imgTr);  
  
 travelListElement.appendChild(travelItem);  
 });  
 }  
  
 *// Добавление обработчиков событий для фильтров* document.getElementById("upcomingFilter").addEventListener("change", *handleFilterChange*);  
 document.getElementById("ongoingFilter").addEventListener("change", *handleFilterChange*);  
 document.getElementById("completedFilter").addEventListener("change", *handleFilterChange*);  
 document.getElementById("postponedFilter").addEventListener("change", *handleFilterChange*);  
 document.getElementById("cancelledFilter").addEventListener("change", *handleFilterChange*);  
  
 window.addEventListener('DOMContentLoaded', function() {  
 const spiderScroll = document.querySelector('.spider-scroll');  
 spiderScroll.addEventListener('scroll', function() {  
 if (spiderScroll.scrollTop > 0) {  
 spiderScroll.style.backgroundColor = '#ffffff';  
 } else {  
 spiderScroll.style.backgroundColor = '#000000';  
 }  
 });  
 });  
  
 const content = document.querySelector('.content');  
  
 content.addEventListener('scroll', function() {  
 content.scrollTop += 10; *// Измените значение, чтобы прокручивать на нужное количество пикселей* });  
  
</script>  
</body>  
</html>

Основная страница(account page):

<!DOCTYPE html>

<html lang="ru">

<head>

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800;900&display=swap"

rel="stylesheet">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Аккаунт</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800;900&display=swap');

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Poppins', sans-serif;

}

.background{

color: black;

background-size: cover;

background-position: center;

background-attachment: fixed;

height: 100%;

width: 100%;

z-index: 1;

}

body {

min-height: 100vh;

}

header {

position: absolute;

top: 0;

left: 0;

width: 100%;

padding: 20px 40px;

display: flex;

justify-content: space-between;

align-items: center;

z-index: 100;

user-select: none;

}

h5 {

color: #fff3cd; /\* Зеленый цвет \*/

text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;

animation: fog 20s;

}

h2{

color: #fff3cd; /\* Зеленый цвет \*/

text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;

}

.logo {

font-size: 1.75em;

color: #fff;

}

.navigation {

display: flex;

align-items: center;

justify-content: center;

gap: 0px;

}

.navigation a {

position: relative;

font-size: 1.25em;

color: #fff;

font-weight: 500;

margin-left: 75px;

padding: 5px 25px;

transition: .5s;

}

.navigation a:hover {

color: #fff3cd;

text-shadow: 0px 0px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;

}

.navigation a span {

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: #ffffff fixed;

border-radius: 50px;

z-index: -1;

transform: scale(0);

opacity: 0;

transition: .5s;

}

.navigation a:hover span {

transform: scale(1);

opacity: 0.25;

}

.parallax {

position: relative;

display: flex;

justify-content: center;

align-items: center;

height: 115vh;

background: url("../css/img/background.png") center center no-repeat;

background-size: cover;

background-position: center;

background-attachment: fixed;

width: 100%;

z-index: -1;

}

#text {

position: absolute;

font-size: 5em;

color: #fff3cd;

text-shadow: 2px 4px 10px #ff7518, 0px 0px 20px #ff7518, 0px 0px 30px #ff7518;

}

.parallax img {

position: absolute;

top: 0;

left: 0;

width: 100%;

pointer-events: none;

}

.parallax img#gate-left {

transform: translateX(-310px);

}

.parallax img#gate-right {

transform: translateX(270px);

;

}

.navigation ul {

display: flex;

list-style: none;

}

.navigation ul li {

margin-left: 40px;

}

.navigation ul li a {

color: #fff;

font-size: 12px;

font-weight: 300;

text-decoration: none;

}

.navigation ul li a:hover {

color: #f1f1f1;

}

.seс {

position: relative;

background: #020912;

padding: 30px 100px;

}

.sec h2{

font-size: 2.5em;

color: #fff;

margin-bottom: 15px;

}

.sec p{

font-size: 1em;

color: #fe3901;

font-weight: 300;

text-align: justify;

}

</style>

</head>

<body>

<section class="parallax">

<h4 id="text">Welcome back!</h4>

<img src="../css/img/tree-left.png" id="tree-left">

<img src="../css/img/tree-right.png" id="tree-right">

<img src="../css/img/gate-left.png" id="gate-left">

<img src="../css/img/gate-right.png" id="gate-right">

<img src="../css/img/grass.png" id="grass">

</section>

<header>

<h5 class="logo"> Profile of mine</h5>

<nav class="navigation">

<a href="#"> travels of mine <span></span></a>

<a href="#">settings<span></span></a>

</nav>

</header>

<section class="sec">

<h2> Рекомендации: </h2>

<p> Ваши рекомендации будут тут(your recommendations)<br><br>

</p>

</section>

<script>

let text = document.getElementById('text');

let treeLeft = document.getElementById('tree-left');

let treeRight = document.getElementById('tree-right');

let gateLeft = document.getElementById('gate-left');

let gateRight = document.getElementById('gate-right');

window.addEventListener('scroll',()=>

{

let value = window.scrollY;

text.style.marginTop = value \* 2 +'px';

treeLeft.style.left = value \* -1.5 +'px';

gateLeft.style.left = value \* 0.5 +'px';

treeRight.style.left = value \* 1.5 +'px';

gateRight.style.left = value \* -0.5 +'px';

}

);

function analyzePreferences() {

// Функция загрузки данных пользователей и предпочтений

function loadUsersData() {

// ваш код загрузки данных пользователей

// Например, можно использовать fetch для получения данных с сервера:

return fetch('/users').then(response => response.json());

}

function loadPreferencesData() {

// ваш код загрузки данных предпочтений

// Например, можно использовать fetch для получения данных с сервера:

return fetch('/preferences').then(response => response.json());

}

// Функция создания матрицы предпочтений пользователя

function createPreferencesMatrix(usersData, preferencesData) {

// ваш код для создания матрицы предпочтений пользователя

// Например, можно использовать объект Map для хранения матрицы предпочтений пользователя:

const preferencesMatrix = new Map();

for (const user of usersData) {

const userPreferences = preferencesData.filter(preference => preference.userId === user.id);

preferencesMatrix.set(user.id, userPreferences);

}

return preferencesMatrix;

}

// Функция расчета сходства между пользователями

function calculateUserSimilarity(preferencesMatrix) {

// ваш код для расчета сходства между пользователями

// Например, можно использовать популярный алгоритм коллаборативной фильтрации - косинусное сходство:

const similarityMatrix = new Map();

for (const [userId1, preferences1] of preferencesMatrix) {

const similarities = new Map();

for (const [userId2, preferences2] of preferencesMatrix) {

const similarity = calculateCosineSimilarity(preferences1, preferences2);

similarities.set(userId2, similarity);

}

similarityMatrix.set(userId1, similarities);

}

return similarityMatrix;

}

// Функция предсказания предпочтений для новых пользователей

function predictPreferences(newUserData, similarityMatrix) {

// ваш код для предсказания предпочтений для новых пользователей

// Например, можно взвешенно усреднить предпочтения похожих пользователей:

const predictedPreferences = new Map();

for (const [newUserId, newUserPreferences] of newUserData) {

const predictedUserPreferences = new Map();

for (const [userId, similarityValues] of similarityMatrix) {

let totalSimilarity = 0;

let weightedPreferencesSum = 0;

for (const [similarUserId, similarity] of similarityValues) {

const similarUserPreferences = preferencesMatrix.get(similarUserId);

if (similarUserPreferences.has(itemId)) {

const preferenceValue = similarUserPreferences.get(itemId);

const weightedPreferenceValue = similarity \* preferenceValue;

totalSimilarity += similarity;

weightedPreferencesSum += weightedPreferenceValue;

}

}

const predictedPreferenceValue = weightedPreferencesSum / totalSimilarity;

predictedUserPreferences.set(itemId, predictedPreferenceValue);

}

predictedPreferences.set(newUserId, predictedUserPreferences);

}

return predictedPreferences;

}

// Функция расчета сходства

function calculateCosineSimilarity(preferences1, preferences2) {

//код для расчета сходства

const dotProduct = preferences1.reduce((sum, value, index) => sum + (value \* preferences2[index]), 0);

const magnitude1 = Math.sqrt(preferences1.reduce((sum, value) => sum + (value \* value), 0));

const magnitude2 = Math.sqrt(preferences2.reduce((sum, value) => sum + (value \* value), 0));

return dotProduct / (magnitude1 \* magnitude2);

}

let resultList = document.createElement('ul'); // Создаем элемент списка

// Пример добавления элементов в список

let item1 = document.createElement('li');

item1.textContent = 'Место 1';

resultList.appendChild(item1);

let item2 = document.createElement('li');

item2.textContent = 'Место 2';

resultList.appendChild(item2);

let item3 = document.createElement('li');

item3.textContent = 'Место 3';

resultList.appendChild(item3);

// Вставляем список на страницу

document.body.appendChild(resultList);

}

analyzePreferences();

</script>

</body>

</html>

1. В этом HTML-коде создается страница "Аккаунт". Она содержит фоновое изображение, шапку с логотипом и навигацией, а также секцию с параллакс-эффектом.

2. В <style> задаются стили для страницы, такие как шрифты, цвета, размеры и смещения элементов.

3. Шапка содержит логотип и навигацию, реализованную с помощью списка и ссылок.

4. При наведении на ссылки в навигации появляется задний фон с эффектом иконки и тени.

5. Секция с параллакс-эффектом содержит текст и изображения ворот, которые смещаются при скроллинге страницы.

6. Код использует внешние файлы шрифтов и изображений.

js-часть:

1. Здесь представлен лишь фрагмент кода- задел для последующих лабораторных работ. Однако нем будут определены несколько функций для анализа пользовательских предпочтений и предсказания предпочтений для новых пользователей.

2. Функция analyzePreferences является основной функцией, которая вызывается в конце скрипта. Она выполняет анализ предпочтений пользователей и отображение результатов на странице.

3. Функция calculateSimilarityMatrix вычисляет матрицу сходства между пользователями на основе их предпочтений. Она возвращает similarityMatrix, который представляет собой Map, где ключом является идентификатор пользователя, а значением - Map с идентификаторами похожих пользователей и их показателями сходства.

4. Функция predictPreferences предлагает рекомендации для новых пользователей на основе сходства. Она возвращает predictedPreferences, который также является Map, где ключом является идентификатор нового пользователя, а значением - Map с рекомендациями для каждого идентификатора элемента.

5. Функция calculateCosineSimilarity вычисляет сходство между двумя векторами предпочтений. Она использует формулу косинусного сходства, где числитель представляет скалярное произведение двух векторов, а знаменатель представляет произведение их длин.

6. Последний блок кода создает список <ul> и добавляет несколько элементов <li> в него. Затем список вставляется на страницу.

7. Функция analyzePreferences вызывает остальные функции и выполняет анализ предпочтений пользователей. Однако, сама логика и детали реализации показаны в отсутствующей части кода.

Возможные дополнения к выше изложенной реализации:

Для функции поиска и бронирования отелей можно дополнительно создать ссылку на форму с полями для ввода даты заезда и выезда, количества гостей и места назначения. При отправке формы можно использовать AJAX для отправки запроса на сервер и получения результатов поиска без перезагрузки страницы.

Для функции подсказок и рекомендаций можно использовать алгоритмы машинного обучения на сервере для анализа предпочтений пользователя и предложения интересных мест. Результаты можно отображать в виде списка или карусели с фотографиями и описанием мест.

Для функции интерактивных карт можно использовать библиотеки JavaScript, такие как Google Maps API или Leaflet, для отображения карты и добавления маркеров, информации о местах и других интерактивных элементов.

Для функции сохранения фотографий и заметок можно использовать базу данных или файловую систему на сервере для хранения фотографий и текстовых данных. Пользователь может загружать фотографии и добавлять заметки в свой профиль, а затем просматривать их в приложении.

Для функции планирования событий и активностей можно создать форму, где пользователь может добавлять события и активности в свой маршрут. Можно использовать календарь или список с возможностью добавления, удаления и редактирования элементов.