Подгузов Денис Ис 24-21

Тема: Использование Fetch API для работы с данными пользователей. Цель: Освоить применение Fetch API для получения, обработки и отображения данных на веб-странице с помощью JavaScript.

### ****Контрольные вопросы****

**Современные технологии в информационных системах**  
К ключевым направлениям развития ИТ относятся:  
облачные сервисы, искусственный интеллект, технологии анализа больших данных (Big Data), Интернет вещей (IoT), блокчейн, а также сфера виртуальной и дополненной реальности (VR/AR). Эти технологии активно внедряются в образование, бизнес и науку.

**Сущность облачных вычислений**  
Облачные технологии позволяют пользователю получать вычислительные ресурсы — хранилища данных, серверы, программное обеспечение — через интернет. При этом нет необходимости устанавливать оборудование и программы локально: всё работает на удалённых серверах.

**Преимущества и риски искусственного интеллекта**

✔ **Преимущества**:  
— автоматизация рутинных процессов  
— повышение точности и скорости анализа данных  
— персонализация услуг  
— рост производительности и эффективности систем

⚠ **Риски**:  
— вероятность ошибок в алгоритмах  
— вытеснение некоторых профессий  
— вопросы этики и ответственности  
— угрозы конфиденциальности и безопасности данных

**Роль Интернета вещей (IoT)**  
IoT объединяет устройства в единую сеть, что позволяет:  
— создавать умные дома и города  
— использовать носимые гаджеты для мониторинга здоровья  
— автоматизировать промышленность и транспорт  
— улучшать контроль ресурсов и окружающей среды

**Почему блокчейн называют “технологией доверия”**  
Блокчейн хранит данные в распределённой системе, использует криптографическую защиту и обеспечивает прозрачность всех операций. Изменить информацию без согласия большинства участников сети практически невозможно, что делает подделку данных крайне сложной.

**Информационная безопасность в образовании и науке**  
Основные меры безопасности:  
— защита персональных данных обучающихся и сотрудников  
— ограничение прав доступа к информации  
— предотвращение кибератак  
— безопасное хранение и передача научных и исследовательских материалов

## Теоретический обзор: Fetch API

Fetch API — современный механизм выполнения HTTP-запросов в JavaScript.  
Он пришёл на замену устаревшему XMLHttpRequest, делая процесс взаимодействия с сервером легче и понятнее.

Основой Fetch API являются **промисы**. Пример типичного запроса:

fetch('https://example.com/data')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Ошибка:', error));

🔹 fetch() возвращает Promise:  
— fulfilled — если запрос успешно выполнен  
— rejected — при сетевой ошибке или недоступности ресурса

Для удобства часто используется async/await, что делает код похожим на синхронный.  
Важно проверять свойство response.ok, чтобы корректно обработать ошибки сервера.

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Список пользователей</title>

<style>

body {

font-family: "Segoe UI", sans-serif;

background: linear-gradient(135deg, #74ebd5, #ACB6E5);

margin: 0;

padding: 20px;

}

h1 {

text-align: center;

color: #222;

text-shadow: 1px 1px 2px #fff;

}

#loading, #error {

font-size: 18px;

margin: 15px 0;

text-align: center;

}

#error {

color: #b30000;

font-weight: bold;

display: none;

}

#usersTable {

width: 90%;

margin: 20px auto;

background: #ffffffcc;

border-collapse: collapse;

box-shadow: 0 0 15px rgba(0,0,0,0.2);

border-radius: 8px;

overflow: hidden;

}

th, td {

padding: 12px 10px;

text-align: left;

}

th {

background: #5568FE;

color: white;

font-weight: bold;

}

tr:nth-child(even) {

background: #f9f9f9;

}

tr:hover {

background: #e1e9ff;

cursor: pointer;

}

button {

display: block;

margin: 25px auto;

padding: 12px 25px;

font-size: 16px;

color: #fff;

background: #5568FE;

border: none;

border-radius: 25px;

cursor: pointer;

transition: 0.3s;

font-weight: bold;

}

button:hover {

background: #3344d0;

transform: scale(1.05);

}

</style>

</head>

<body>

<h1>Список пользователей</h1>

<div id="loading">Загрузка...</div>

<div id="error">Ошибка при получении данных</div>

<table id="usersTable">

<thead>

<tr>

<th>ID</th>

<th>Имя пользователя</th>

<th>Email</th>

<th>Город</th>

</tr>

</thead>

<tbody>

</tbody>

</table>

<button id="refreshBtn">Обновить данные</button>

<script>

const usersTableBody = document.querySelector('#usersTable tbody');

const loadingDiv = document.getElementById('loading');

const errorDiv = document.getElementById('error');

const refreshBtn = document.getElementById('refreshBtn');

// Генерация рандомных данных

const names = ["Alex", "Denis", "Michael", "Nurlan", "Artem", "Oleg", "Timur", "David", "Max", "Daniel"];

const cities = ["Almaty", "Astana", "Shymkent", "Karaganda", "Aktobe", "Moscow", "Berlin", "Paris", "Dubai", "Los Angeles"];

function getRandomEmail(name) {

const domains = ["gmail.com", "yahoo.com", "mail.ru", "hotmail.com"];

return `${name.toLowerCase()}${Math.floor(Math.random() \* 1000)}@${domains[Math.floor(Math.random() \* domains.length)]}`;

}

async function loadUsers() {

usersTableBody.innerHTML = '';

errorDiv.style.display = 'none';

loadingDiv.style.display = 'block';

try {

const users = [];

for (let i = 1; i <= 10; i++) {

const randomName = names[Math.floor(Math.random() \* names.length)];

const randomCity = cities[Math.floor(Math.random() \* cities.length)];

const randomEmail = getRandomEmail(randomName);

users.push({

id: i,

username: randomName,

email: randomEmail,

city: randomCity

});

}

users.forEach(user => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${user.id}</td>

<td>${user.username}</td>

<td>${user.email}</td>

<td>${user.city}</td>

`;

usersTableBody.appendChild(row);

});

} catch (error) {

errorDiv.style.display = 'block';

console.error('Ошибка при получении данных:', error);

} finally {

loadingDiv.style.display = 'none';

}

}

loadUsers();

refreshBtn.addEventListener('click', () => {

loadUsers();

});

</script>

</body>

</html>

