**Работа с API и управление данными**

Современные веб-приложения часто взаимодействуют с внешними источниками данных, такими как серверы, базы данных или сторонние сервисы. Для этого используются API (Application Programming Interface) — интерфейсы, позволяющие получать и отправлять данные. В этой лекции мы разберем основы работы с API, способы отправки запросов и обработки ответов, а также принципы управления данными на стороне клиента.

**Что такое API**

API — это набор методов и правил, позволяющих взаимодействовать с программным обеспечением или сервисом. В контексте веб-разработки чаще всего используется **REST API** — архитектурный стиль, основанный на передаче данных через HTTP-запросы.

Основные виды HTTP-запросов:

* **GET**: получение данных.
* **POST**: отправка данных.
* **PUT/PATCH**: обновление данных.
* **DELETE**: удаление данных.

API обычно возвращает данные в формате **JSON (JavaScript Object Notation)**, который представляет собой удобную для чтения и обработки структуру.

**Отправка запросов с помощью fetch**

Для работы с API в JavaScript используется встроенный метод fetch. Этот метод позволяет отправлять запросы и обрабатывать ответы.

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

  .then(response => response.json()) // Преобразование ответа в JSON

  .then(data => console.log(data)) // Работа с полученными данными

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

//fetch возвращает Promise, что позволяет обрабатывать данные асинхронно. Чтобы сделать код более читабельным, используется async/await:

async function fetchData() {

  try {

    const response = await fetch(

      'https://api.example.com/data',

    );

    const data = await response.json();

    console.log(data);

  } catch (error) {

    console.error('Ошибка:', error);

  }

}

fetchData();

**Работа с данными**

После получения данных их часто нужно обработать, например, отобразить на странице, сохранить или передать другим модулям.

Пример обработки данных:

1. Отправить запрос к API.
2. Получить список пользователей.
3. Отобразить имена пользователей в виде списка на веб-странице.

**Пример: отображение данных из API**

Приведем пример работы с реальным API, где мы получаем данные о пользователях и выводим их на страницу.

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Работа с API</title>

</head>

<body>

  <h1>Список пользователей</h1>

  <ul id="userList"></ul>

  <script>

    async function loadUsers() {

      try {

        // Отправка GET-запроса к API

        const response = await fetch("https://jsonplaceholder.typicode.com/users");

        const users = await response.json();

        // Найти элемент списка

        const userList = document.getElementById("userList");

        // Отобразить пользователей

        users.forEach(user => {

          const listItem = document.createElement("li");

          listItem.textContent = `${user.name} (${user.email})`;

          userList.appendChild(listItem);

        });

      } catch (error) {

        console.error("Ошибка при загрузке данных:", error);

      }

    }

    // Вызов функции загрузки данных

    loadUsers();

  </script>

</body>

</html>