**Перед началом урока проверяем у детей слова с прошлого урока**

|  |  |
| --- | --- |
| **Английское слово** | **Перевод на русский** |
| function | функция — набор команд |
| createElement | создать элемент |
| appendChild | добавить дочерний элемент |
| textContent | текстовое содержимое |
| getElementById | получить элемент по ID |

**Цель урока**

На этом уроке мы создадим **сервер на JavaScript** с использованием библиотеки **Express**. Сервер будет обрабатывать запросы и обмениваться данными между компьютерами в локальной сети.

**Что такое IP-адрес?**

### Каждому компьютеру в сети присваивается уникальный IP-адрес — это как домашний адрес в Интернете. Он нужен, чтобы другие устройства знали, куда отправлять информацию.

### Виды IP-адресов

* **Локальный IP** — это адрес компьютера внутри вашей локальной сети (в классе, дома, школе).  
  Обычно начинается с 192.168...
* **Глобальный IP** — это адрес компьютера или роутера, видимый в Интернете.

**Как узнать IP-адрес на компьютере?**

Локальный IP (внутри сети)

1. Нажмите **Пуск** и откройте **Панель управления**.
2. Перейдите в раздел **Сеть и Интернет** → **Центр управления сетями и общим доступом**.
3. Справа нажмите на активное подключение (например, "Беспроводное сетевое соединение" или "Ethernet").
4. В открывшемся окне нажмите кнопку **Сведения...**.
5. Найдите строку **IPv4-адрес** — это и есть локальный IP-адрес компьютера.  
   Пример: 192.168.1.104

Иногда в школе вкладка «Сеть и Интернет» может быть недоступна. В таком случае можно посмотреть IP-адрес через консоль.

1. Нажмите клавиши **Win + R**.
2. В появившемся окне введите cmd и нажмите **Enter** — откроется чёрное окно (Консоль).
3. Введите команду: ipconfig
4. Найдите строку **IPv4-адрес**. Это и есть локальный IP.

Глобальный IP (в Интернете)

1. Откройте браузер.
2. Перейдите на сайт: <https://2ip.ru/>
3. На главной странице вы увидите ваш **внешний (глобальный) IP-адрес**.

**Что мы создаём?**

На **каждом компьютере** будет работать **локальный сервер**, написанный на Node.js. Этот сервер будет отвечать на запросы от других компьютеров в сети.

**Что такое API?**

**API (Application Programming Interface)** — это способ, с помощью которого программы «разговаривают» между собой.  
У сервера есть специальные **адреса** (их называют маршруты или endpoints), по которым можно отправить запрос и получить ответ.

### 🚀 Наш сервер будет уметь:

1. **GET /profile**
   * Возвращает **тестовую картинку**, а также **фамилию, имя и описание**.
   * Это нужно, чтобы проверить: работает ли сервер на этом компьютере.
   * 📌 Используется как «пинг».
2. **POST /upload**
   * Принимает **картинку от пользователя** и сохраняет её на сервере.
   * То есть с другого компьютера можно загрузить файл.
3. **GET /images**
   * Отдаёт **список всех картинок**, которые загружены на сервер.
   * Каждый сможет посмотреть, что уже добавлено.

### ☕ Аналогия с кафе (для объяснения детям)

* **Клиент** (пользователь, фронтенд) — это как человек в кафе. Он делает заказ (например, загрузить фото или посмотреть список).
* **Официант** (HTTP-запрос) — приносит заказ на кухню и возвращает результат.
* **Кухня** (сервер на Node.js) — готовит еду, то есть выполняет запросы: ищет картинку, сохраняет её или показывает список.
* **Меню** (API) — это список доступных «блюд» — то, что можно «заказать» у сервера.

Примеры:

* GET /profile → «Принеси пробник» (сервер отдаёт картинку + данные о владельце).
* POST /upload → «Я хочу добавить новое блюдо» (загрузить картинку).
* GET /images → «Покажи всё меню» (список всех загруженных картинок).

👨‍🏫 **Совет преподавателю:**  
Объясните детям, что **API — это как меню**: нельзя заказать то, чего нет в меню. Сервер понимает только те «команды», которые в нём прописаны.

**Чтобы начать работу:**

1. Зайдите в папку проекта и нажмите правой кнопкой мыши → Открыть с помощью Code или запустить редактор кода, в меню сверху выберите **File → Open Folder...** (или **Файл → Открыть папку...**) -> Найдите нужную папку с проектом и нажмите **"Выбрать папку" / "Select Folder"**.
2. Откройте **терминал** в редакторе Visual Studio Code. (Нажмите **Ctrl + Ё** или *View → Terminal*).
3. Введите команду: ***npm init -y***.

Эта команда создаст файл package.json — в нём будут храниться все настройки проекта и список используемых библиотек.

1. Установите нужные модули: Введите команду ***npm install express multer cors***

**Объяснение:**

**express** — это основа сервера: с его помощью мы создаём маршруты и обрабатываем запросы.

**multer** — отвечает за загрузку файлов (например, картинок).

**cors** — позволяет другим компьютерам обращаться к серверу (без него браузер может блокировать такие обращения).

Создаем файл server.js

// Загружаем библиотеки

const express = require('express');   // Сам сервер

const cors = require('cors');         // Чтобы разрешить запросы с других адресов

const app = express(); // Создаём приложение express

const PORT = 3000;     // На каком порту будет работать сервер

// Включаем CORS (разрешаем фронту обращаться к серверу)

app.use(cors());

// Запускаем сервер

app.listen(PORT, () => {

  console.log(`Сервер запущен на http://localhost:${PORT}`);

});

**Ответы сервера на маршруты**

На текущем этапе сервер запускается и выводит сообщение в консоли о том, что он работает. Однако взаимодействовать с ним пока невозможно — он не обрабатывает никакие входящие HTTP-запросы.

Чтобы реализовать базовое взаимодействие, необходимо определить маршруты (routes). Это специальные пути, по которым клиент (например, браузер) может отправить запрос на сервер, а сервер — вернуть ответ.

// Загружаем библиотеки

const express = require('express');   // Сам сервер

const cors = require('cors');         // Чтобы разрешить запросы с других адресов

const app = express(); // Создаём приложение express

const PORT = 3000;     // На каком порту будет работать сервер

// Включаем CORS (разрешаем фронту обращаться к серверу)

app.use(cors());

//первый роут

app.get('/hello', (req, res) => {

  res.send('Привет от сервера'); // Отправляем текст в ответ

});

// Запускаем сервер

app.listen(PORT, () => {

  console.log(`Сервер запущен на http://localhost:${PORT}`);

});

После запуска сервера и перехода в браузере по адресу <http://localhost:3000/hello> , пользователь увидит сообщение **«Привет от сервера»**.

**Передача структурированных данных (JSON)**

Следующий шаг — отправка от сервера не текста, а **структурированных данных** в формате JSON. Это стандартный способ обмена данными между клиентом и сервером.

// res.json(...) — метод Express, который отправляет объект в формате JSON.

// Такой формат удобно обрабатывать на стороне клиента — JavaScript может работать с ним как с обычным объектом

app.get('/profile, (req, res) => {

  res.json({

    firstname: 'Вася',

    lastname: 'Пупкин'

  });

});

Можно также проверить вывод данных перейдя по адресу <http://localhost:3000/about> .

**Понимание IP-структуры в локальной сети**

Прежде чем опрашивать все компьютеры в сети, необходимо понять, как устроен IP-адрес в локальной сети и как определить общую часть.

IP-адрес состоит из четырёх чисел, разделённых точками. Пример: 192.168.1.45. В этом примере:

* 192.168.1 — это **подсеть** (общая часть IP-адреса), одинаковая для всех компьютеров в классе;
* 45 — уникальный номер устройства в этой сети.

Чтобы это стало наглядным, можно попросить нескольких учащихся посмотреть и назвать свои IP-адреса. Это позволит им самостоятельно увидеть, что первые три числа у всех совпадают. Так они поймут, что компьютеры находятся в одной локальной сети, и можно перебрать адреса с одинаковым началом.

**Задача: "пинговать" все возможные адреса**

Мы будем отправлять запрос на каждый адрес, добавляя к нему путь /about. Если сервер на этом IP работает — он ответит, и мы вызовем функцию createWindow(...).

В файле script.js вносим следующие изменения