**ВСТУПИТЕЛЬНАЯ ПРОВЕРКА**

**УСТАНОВЛЕН ЛИ Visual Studio Code?**

### Как проверить:

1. Откройте **меню "Пуск"** (или нажмите клавишу Windows).
2. Введите в поиск: Visual Studio Code.
3. Если в результатах появится программа с таким названием — значит Visual Studio Code установлен.

### Как установить Visual Studio Code:

1. Перейдите на официальный сайт: <https://code.visualstudio.com>
2. Нажмите **Download for Windows**.
3. Скачайте установочный файл и **запустите его**.
4. Следуйте инструкциям установщика. При установке желательно включить опцию **"Добавить в контекстное меню"** и **"Добавить в PATH"** — это упростит запуск.

**УСТАНОВЛЕН ЛИ NODE.JS?**

**Как проверить:**

1. Откройте терминал.
2. Введите команду: node -v

Если вы увидите номер версии, например v18.19.1, то всё в порядке.

Если вы получите ошибку вроде **node: command not found**, значит Node.js не установлен.

**Как установить Node.js:**

1. Перейдите на сайт: <https://nodejs.org>
2. Скачайте **LTS-версию** (она стабильнее и подходит для обучения).
3. Установите Node.js, следуя инструкциям на экране.
4. После установки снова откройте терминал и введите node -v — теперь всё должно работать.

**Запустить финальный код на всех ПК для устранения потенциальных проблем.**

Для этого нужно скачать копию проекта на все пк. Для этого:

1. Перейдите на страницу проекта: <https://github.com/Umbomax/tutor_fullstackapp>.
2. Нажмите на зелёную кнопку **Code**.
3. Выберите пункт **Download ZIP**.
4. После скачивания распакуйте архив в удобное для вас место на компьютере

Далее откройте терминал в папке проекта и введите команду npm install. После установки зависимостей запустите сервер командой node server.js. Если всё настроено правильно, вы увидите сообщение вроде: Сервер запущен на порту 3000.

После запуска сервера на всех ПК можно запустить на одном из ПК файл index.html, изменить IP-адреса и загрузить на разные ПК изображения.

**Введение**

Мы будем создавать простое **fullstack-приложение**, которое будет отображать **компьютеры в сети** и показывать, какие изображения загружены на каждом. Это приложение позволит нам обмениваться файлами через локальную сеть.

Но перед тем, как перейти к сложным вещам, давайте начнём с самого простого — **одного "окна" с IP-адресом и изображением**. Сначала мы сделаем его вручную с помощью HTML и CSS, а потом научимся создавать такие окна с помощью JavaScript.

Код для файла index.html

<!DOCTYPE html>

<html lang="ru">

<head>

  <meta charset="UTF-8" />

  <title>Локальная сеть - Учебный проект</title>

  <link rel="stylesheet" href="style.css" />

</head>

<body>

  <h1>Компьютеры в сети</h1>

    <div id="windows-container">

        <div class="window">

            <img src="путь или ссылка на любое изображение" alt="Preview" />

            <div class="ip-address">192.168.1.101</div>

          </div>

    </div>

  <script src="script.js"></script>

</body>

</html>

Код файла style.css

body {

    background-color: #f9f9f9;

    padding: 30px;

}

h1 {

    margin-bottom: 20px;

}

.window {

    width: 220px;

    padding: 15px;

  display: inline-block;

    background-color: #ffffff;

    border: 3px solid #437dda;

    border-radius: 12px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    text-align: center;

    cursor: pointer;

    margin: 10px;

}

.window img {

    width: 180px;

    height: 130px;

    object-fit: cover;

    border-radius: 6px;

    margin-bottom: 10px;

}

.ip-address {

    font-weight: bold;

    color: #333;

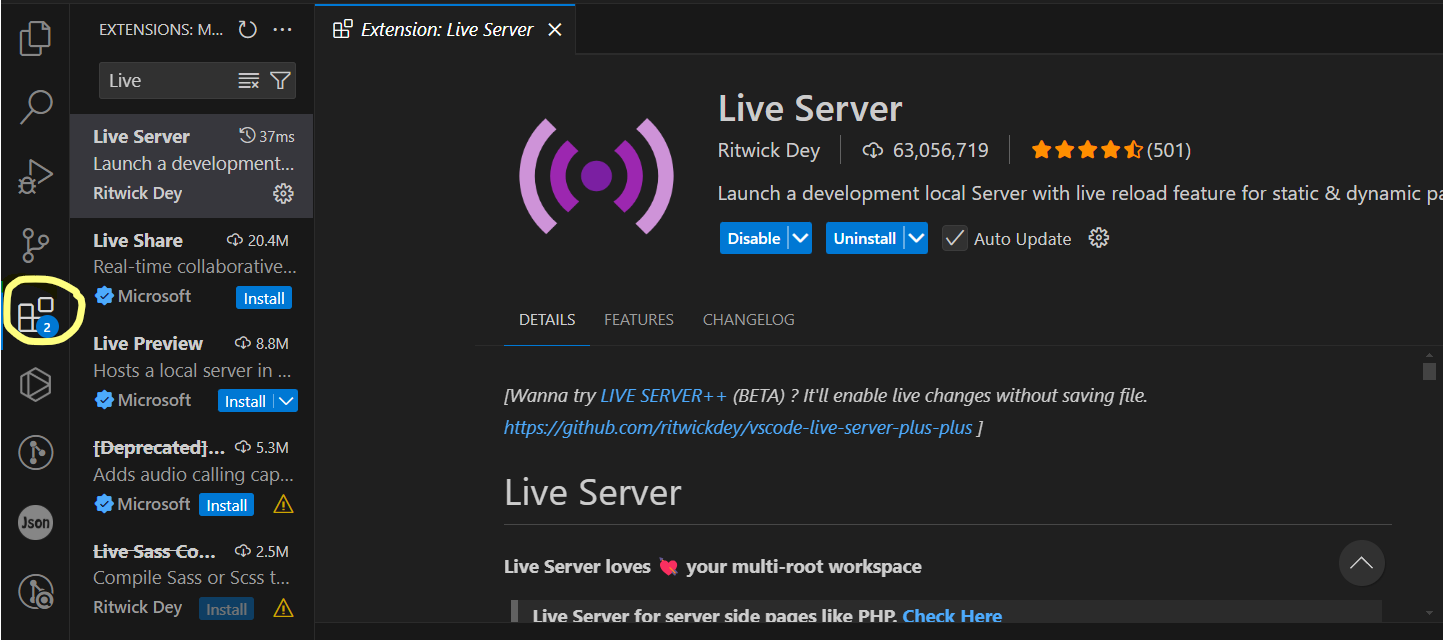
    font-size: 16px;

}

Для удобного наблюдения за изменениями на странице в режиме реального времени рекомендуется установить расширение Open Live Server. Оно позволяет автоматически открывать ваш HTML-файл в браузере и обновлять его при каждом сохранении.

Чтобы изменения отображались сразу:

Откройте вкладку Extensions (Расширения) в редакторе Visual Studio Code (слева).



Найдите и установите расширение Live Server.

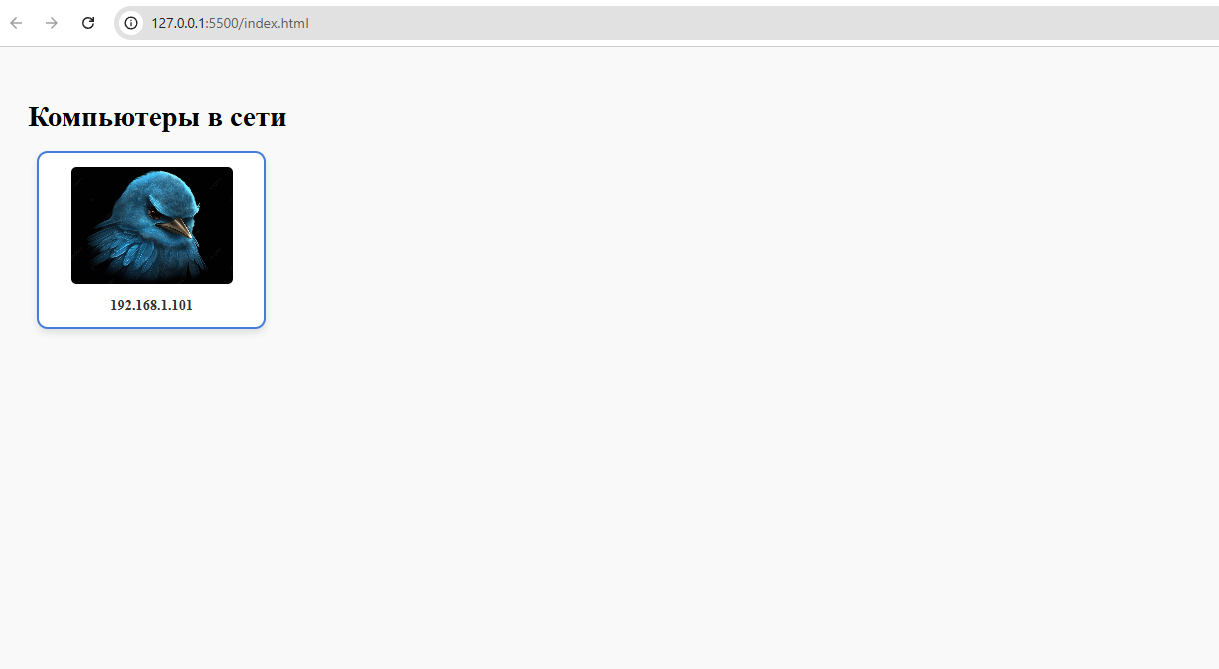
После установки щёлкните правой кнопкой мыши по файлу index.html и выберите "Open with Live Server" — откроется окно браузера, в котором будет отображаться ваша страница.

Включите автосохранение, чтобы каждое изменение файла автоматически отображалось в браузере. Для этого:

Откройте меню File (Файл) → Auto Save (Автосохранение) → Убедитесь, что стоит галочка рядом с этим пунктом.

Теперь вы можете видеть все изменения без необходимости вручную обновлять страницу.

Получается следующая страница:



Мы создали одно окно с помощью HTML и CSS. Мы делаем сайт, чтобы обмениваться изображениями между компьютерами, а это значит, что таких окон у нас будет много — для каждого компьютера в нашей локальной сети.

Сколько окон нам может понадобится мы не знаем! Сегодня у нас в классе 5 компьютеров, а завтра может быть 12. Поэтому писать вручную 12 одинаковых блоков — это не только скучно, но и неудобно.

Нам нужно научиться создавать такие окна автоматически с помощью JavaScript. Он сможет сам создать нужное количество окон, когда мы ему скажем IP-адреса и адреса картинок.

**Создаём окно вручную (без функции)**

Прежде чем писать функцию, давайте шаг за шагом создадим одно «окно» вручную. Это поможет лучше понять, как работают команды JavaScript.

**1. Находим контейнер, куда будем добавлять окно**

const container = document.getElementById("windows-container");

Этот элемент уже есть в HTML. Мы будем вставлять «окна» внутрь него.

2. Создаём блок-окно (div):

const windowDiv = document.createElement("div");

windowDiv.className = "window"; // как будто написали <div class="window">

3. Добавляем изображение:

const img = document.createElement("img");

img.src = " путь или ссылка на любое изображение" // путь к изображению

img.alt = "Preview"; // текст, если картинка не загрузится

windowDiv.appendChild(img); // добавляем изображение внутрь блока окна

Задание для детей

*Теперь попробуйте сами написать код, который добавляет подпись с IP-адресом под изображением.*

*Мы используем textContent, чтобы добавить текст в элемент:*

const ipDiv = document.createElement("div"); // создаём новый div

ipDiv.className = "ip-address"; // задаём класс

ipDiv.textContent = "192.168.1.103"; // пишем текст внутрь

windowDiv.appendChild(ipDiv); // добавляем подпись в окно

4. Добавляем окно на страницу:

container.appendChild(windowDiv);

**Итог**

Теперь у нас на странице появилось окно с картинкой и подписью IP-адреса — и всё это мы сделали вручную, шаг за шагом.

Сейчас напишем функцию, которая будет делать это автоматически для любого IP и картинки.

 // 1. Функция создания окна

 function createWindow(ip, imageUrl) /\* (айпи и ссылка это аргументы функции)\*/ {

    // 2. Находим контейнер на странице, куда будем вставлять окно

    const container = document.getElementById('windows-container');

    // 3. Создаём div — это наш основной блок окна

    const windowDiv = document.createElement('div');

    windowDiv.className = 'window'; //задаем элементу класс как в html

    // 4. Создаём тег img для картинки

    const img = document.createElement('img');

    img.src = imageUrl;//задаем элементу src как в html

    img.alt = 'Preview';

    // 5. Создаём подпись с IP

    const ipDiv = document.createElement('div');

    ipDiv.className = 'ip-address';

    ipDiv.textContent = ip;

    // 6. Вставляем картинку и IP в окно

    windowDiv.appendChild(img);

    windowDiv.appendChild(ipDiv);

    // 7. Вставляем окно в контейнер на странице

    container.appendChild(windowDiv);

  }

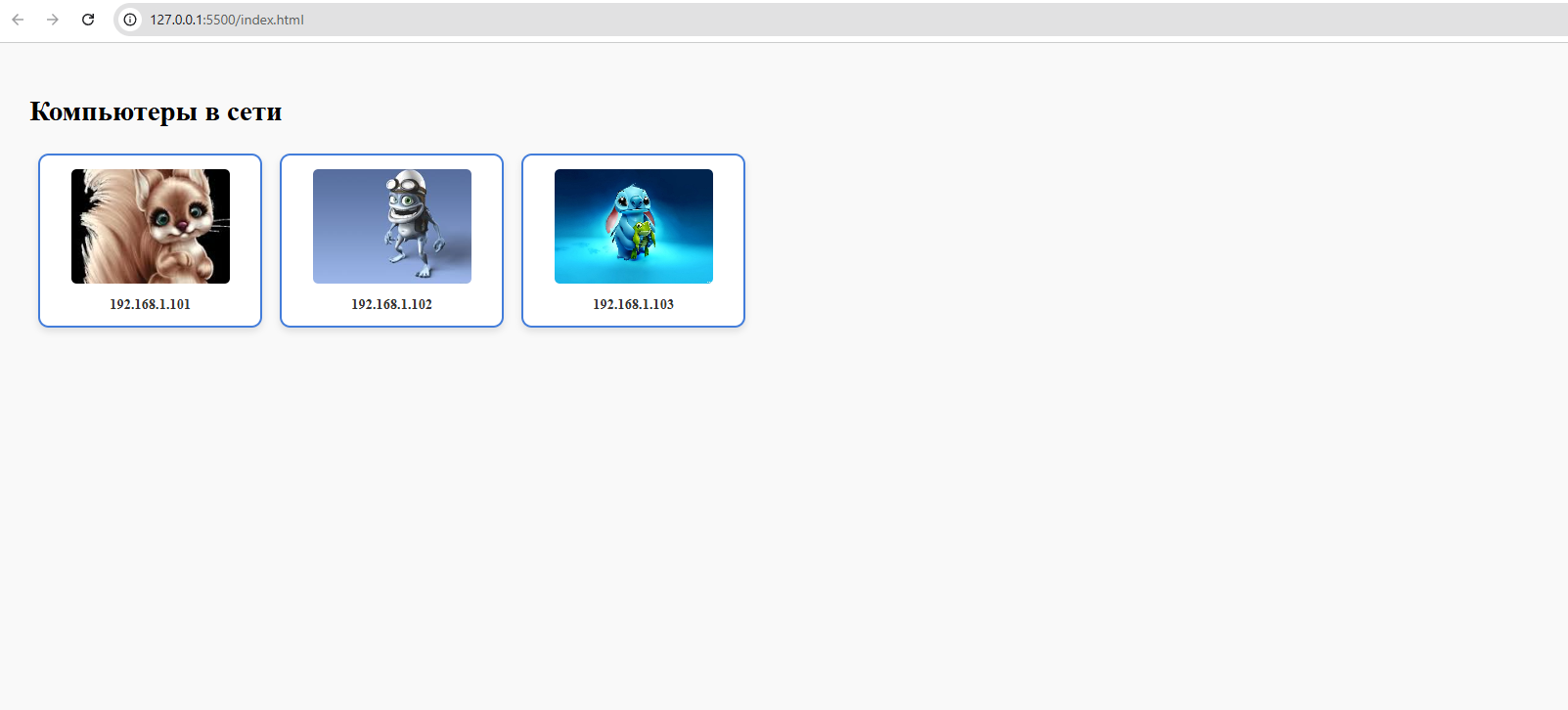
  // 8. Пример: создаём три окна

  createWindow('192.168.1.101', "путь или ссылка на любое изображение");

  createWindow('192.168.1.102', "путь или ссылка на любое изображение");

  createWindow('192.168.1.103', "путь или ссылка на любое изображение");

**В файле index.html удаляем код окна и смотрим на результат.**



Во время написания кода полезно проверять, как выглядят создаваемые элементы. Для этого можно вывести переменную **windowDiv** в консоль, добавив в код строку:

    console.log(windowDiv);

Чтобы увидеть результат:

1. Откройте вашу страницу в браузере.
2. Нажмите клавишу **F12**, чтобы открыть инструменты разработчика.
3. Перейдите во вкладку **Console (Консоль)**.

В этой вкладке будет отображаться информация о переменной windowDiv, включая её структуру и содержимое. Это помогает понять, правильно ли работает ваш код и что именно добавляется на страницу.

Пример постепенного написания кода с выводом переменной в консоль:

  function createWindow(ip, imageUrl){

    const container = document.getElementById("windows-container");

    const windowDiv = document.createElement("div");

    windowDiv.className = "window"; //задаем элементу класс как в html

    console.log(windowDiv);

    container.appendChild(windowDiv);

}

createWindow("192.168.1.103", "путь или ссылка на любое изображение");

В HTML-структуре один элемент может содержать другие — они называются **дочерними элементами**. Слово **child** в переводе с английского означает **«ребёнок»**. В коде мы используем команду appendChild. Команда appendChild добавляет один элемент внутрь другого — как будто «прикрепляет ребёнка к родителю».

Например, изображение и подпись с IP-адресом находятся *внутри* блока-окна. Это значит, что они являются дочерними элементами по отношению к окну.

**Важно**, чтобы дети лучше запоминали новые слова. Поэтому в начале каждого урока преподавателю рекомендуется проверять знание английских слов из словаря.

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