

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем  
Факультет автоматики и вычислительной техники  
Кафедра электронных вычислительных машин

Дата сдачи на проверку:

«\_\_» \_\_\_\_\_ 2025 г.

Проверено:

«\_\_» \_\_\_\_\_ 2025 г.

Разработка клиент–серверного веб-приложения на HTML, CSS, JavaScript и  
Express.js.

Отчёт по лабораторной работе №2  
по дисциплине  
«Технологии Программирования»

Разработал студент гр. ИВТб-2301-05-00

\_\_\_\_\_/Черкасов А. А./

(подпись)

Преподаватель

\_\_\_\_\_/Пащенко Д. Э./

(подпись)

Киров  
2025

## Цели лабораторной работы

- изучить структуру клиент–серверных веб-приложений;
- освоить работу с Express.js и обработку POST-запросов;
- закрепить навыки раздельного использования HTML, CSS и JavaScript;
- научиться принимать и обрабатывать данные, отправленные из формы.

## Задание

Создать веб-приложение, работающее по архитектуре «клиент–сервер», удовлетворяющее требованиям:

1. На стороне клиента реализовать HTML-страницу с формой отправки данных.
2. Стили разместить в отдельном файле CSS, логику обработки формы — в отдельном JS.
3. На стороне сервера реализовать обработку POST-запроса с использованием Express.js.
4. Сервер должен принимать данные, логировать их и возвращать ответ клиенту.
5. Организовать вывод всех полученных данных в виде массива.
6. Файлы клиента должны раздаваться через статическую директорию.

## Ход выполнения работы

### 1. Создание структуры проекта

Проект был создан в соответствии с типовой структурой веб-приложения:

- `index.html` — основная страница с формой;
- `style.css` — стили оформления;
- `script.js` — логика отправки данных;
- `server.js` — сервер на Express.js;
- статические файлы размещены в папке `/public`.

### 2. Реализация клиентской части

Была разработана HTML-форма, содержащая несколько полей ввода и кнопку отправки. Все стили вынесены в отдельный файл CSS. JavaScript осуществляет отправку данных методом POST и выводит ответ сервера.

### 3. Реализация серверной части

Сервер был создан на Express.js. Были подключены модули:

- `express` — основной фреймворк;
- `body-parser` — обработка POST-запросов.

На сервере была реализована логика:

- раздача статических файлов;
- логирование всех входящих запросов (данные формы, `query`, `headers`);
- сохранение всех полученных данных в массив;
- ответ клиенту после успешного получения данных;
- вывод массива по запросу `/list`.

Также дополнительно был реализован расширенный лог всего, что происходит в сервере: тело запроса, параметры, заголовки, время получения.

## 4. Тестирование работы приложения

После запуска сервера через:

```
bun server.js
```

или

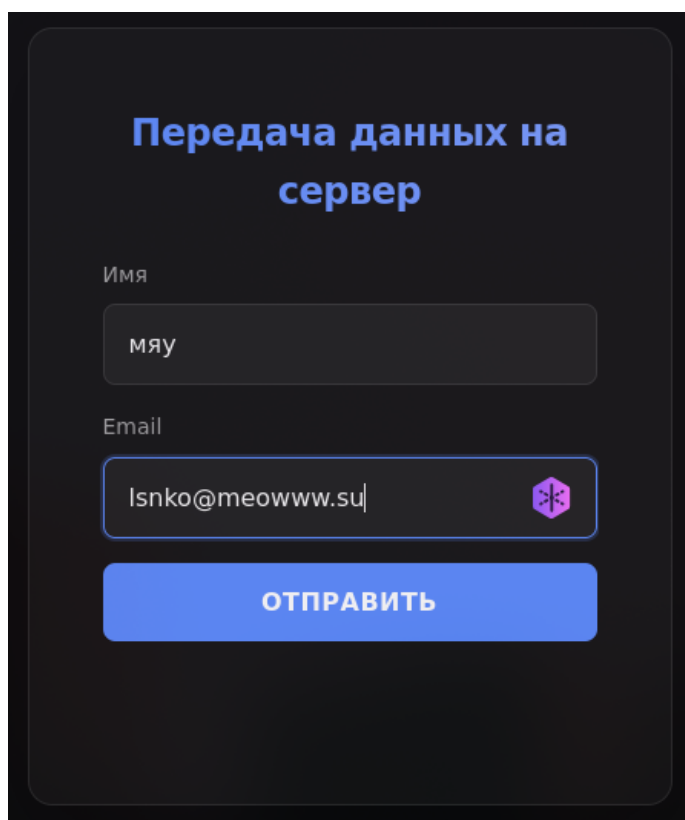
```
node server.js
```

Приложение было открыто в браузере по адресу:

`http://localhost:8080`

Поля формы были заполнены, данные успешно отправлены и отобразились в консоли, а затем были возвращены пользователю.

## 5. Скриншот формы



**Передача данных на сервер**

Имя

мяу

Email

lsnko@meowwww.su

**ОТПРАВИТЬ**

Рисунок 1 — Форма

## Вывод

В ходе выполнения лабораторной работы было разработано полноценное клиент–серверное веб-приложение. Была изучена структура веб-проекта, реализован сервер на Express.js, выполнено разделение клиентского кода на HTML, CSS и JavaScript.

Были выполнены все требования задания: форма корректно отправляет данные, сервер принимает их, логирует, сохраняет и возвращает ответ клиенту.

Работа позволила закрепить знания по основам веб-разработки и принципам построения архитектуры «клиент–сервер».

## Приложение А1. Исходный код index.html

```
<!doctype html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Лабораторная форма</title>
  <link rel="icon" href="pics/handsome.png" type="image/png">
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <h1 class="title">Meme Repo</h1>

    <form id="form" enctype="multipart/form-data">
      <div class="form-group">
        <label class="form-label">Имя</label>
        <input class="form-control" name="name" required>
      </div>

      <div class="form-group">
        <label class="form-label">Email</label>
        <input class="form-control" name="email" type="email" required>
      </div>

      <div class="form-group">
        <label class="form-label">Мем</label>
        <input class="form-control" type="file" name="meme" accept="image/*" required>
      </div>

      <button class="btn">Отправить</button>
    </form>

    <div id="response" class="response"></div>
  </div>

  <div class="spinning-meme"></div>
```

```

<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>
<div class="spinning-meme"></div>

<script src="script.js"></script>
</body>
</html>

```

## Приложение А2. Исходный код script.js

```

const form = document.getElementById('form');
const responseBox = document.getElementById('response');
const submitBtn = form.querySelector('.btn');

const handleSubmit = async (e) => {
  e.preventDefault();

  const data = new FormData(form);
  showLoadingState();

  try {
    const res = await fetch('/', {
      method: 'POST',
      body: data
    });

    const text = await res.text();
    showResponse(`Сервер ответил: ${text}`);
    form.reset();
  }

```

```

    } catch (error) {
      showResponse(`Ошибка: ${error.message}`);
    } finally {
      resetButtonState();
    }
  };

```

```

const showLoadingState = () => {
  submitBtn.disabled = true;
  submitBtn.textContent = 'Отправка...';
};

```

```

const showResponse = (message) => {
  responseBox.textContent = message;
  responseBox.classList.add('show');

  setTimeout(() => {
    responseBox.classList.remove('show');
  }, 5000);
};

```

```

const resetButtonState = () => {
  submitBtn.disabled = false;
  submitBtn.textContent = 'Отправить';
};

```

```

// Fetch images and randomly assign to spinning memes
fetch('/images')
  .then(res => res.json())
  .then(images => {
    // Randomly position spinning memes without overlap
    const spinningMemes = document.querySelectorAll('.spinning-meme');
    const placedPositions = [];

    spinningMemes.forEach((meme, index) => {
      let randomTop, randomLeft;

```

```

let attempts = 0;
const maxAttempts = 100;

do {
  randomTop = Math.random() * 70 + 10; // 10% to 80% to leave space
  randomLeft = Math.random() * 70 + 10; // 10% to 80%
  attempts++;
} while (isOverlapping(randomTop, randomLeft) && attempts < maxAttempts);

placedPositions.push({ top: randomTop, left: randomLeft });
meme.style.top = randomTop + '%';
meme.style.left = randomLeft + '%';

// Assign random image
if (images.length > 0) {
  const randomImage = images[Math.floor(Math.random() * images.length)];
  meme.style.backgroundImage = `url('${randomImage}')`;
}
});
})
.catch(err => console.error('Failed to fetch images:', err));

function isOverlapping(top, left) {
  const size = 15; // Approximate size in % (100px at 100% would be ~15% depending on screen)
  for (const pos of placedPositions) {
    if (Math.abs(pos.top - top) < size && Math.abs(pos.left - left) < size) {
      return true;
    }
  }
  return false;
}

form.addEventListener('submit', handleSubmit);

```

## Приложение А3. Исходный код style.css

```
:root {
```

```

/* Y2K Color palette */
--bg-primary: linear-gradient(135deg, #ff00ff, #00ffff, #ffff00);
--bg-secondary: linear-gradient(45deg, #ff1493, #00bfff, #32cd32);
--bg-tertiary: linear-gradient(90deg, #ff69b4, #1e90ff, #00ff7f);
--accent-primary: #ff00ff;
--accent-secondary: #00ffff;
--text-primary: #ffffff;
--text-secondary: #ffff00;
--border-color: #ffffff;
--glow-color: #ff00ff;

/* Typography */
--font-main: 'Arial Black', 'Impact', sans-serif;
--font-size-base: 1rem;
--font-size-lg: 1.25rem;
--font-size-xl: 1.5rem;

/* Spacing */
--space-sm: 0.5rem;
--space-md: 1rem;
--space-lg: 1.5rem;
--space-xl: 2rem;
--space-xxl: 3rem;

/* Border radius */
--radius-sm: 0px;
--radius-md: 0px;
--radius-lg: 0px;
--radius-xl: 0px;

/* Shadows */
--shadow-sm: 0 0 10px var(--glow-color);
--shadow-md: 0 0 20px var(--glow-color);
--shadow-lg: 0 0 30px var(--glow-color);
--shadow-xl: 0 0 40px var(--glow-color);

/* Transitions */

```

```

--transition-fast: all 0.15s ease;
--transition-normal: all 0.25s ease;
--transition-slow: all 0.35s ease;
}

/* Base styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: var(--font-main);
  background: linear-gradient(135deg, #ff00ff, #00ffff, #ffff00);
  background-size: 400% 400%;
  animation: gradientShift 5s ease infinite;
  color: var(--text-primary);
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  padding: var(--space-xl);
  line-height: 1.6;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

@keyframes gradientShift {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}

/* Container styles */
.container {
  width: 100%;

```

```

max-width: 420px;
background: linear-gradient(45deg, #ff1493, #00bfff, #32cd32);
background-size: 300% 300%;
animation: containerGlow 3s ease-in-out infinite alternate;
padding: var(--space-xxl);
border-radius: var(--radius-xl);
box-shadow: var(--shadow-xl);
border: 2px solid var(--border-color);
transition: var(--transition-normal);
}

.container: hover {
  box-shadow: var(--shadow-lg);
  animation-duration: 1s;
}

@keyframes containerGlow {
  0% { box-shadow: 0 0 20px var(--glow-color); }
  100% { box-shadow: 0 0 40px var(--glow-color); }
}

/* Title styles */
.title {
  text-align: center;
  margin-bottom: var(--space-lg);
  font-size: var(--font-size-xl);
  font-weight: 900;
  letter-spacing: 0.1em;
  text-transform: uppercase;
  color: var(--text-primary);
  text-shadow: 0 0 10px var(--glow-color), 0 0 20px var(--glow-color), 0 0 30px var(--glow-color);
  animation: titleBlink 2s ease-in-out infinite alternate;
}

@keyframes titleBlink {
  0% { text-shadow: 0 0 10px var(--glow-color); }
  100% { text-shadow: 0 0 20px var(--glow-color), 0 0 30px var(--glow-color); }
}

```

```

}

/* Form styles */
form {
  display: flex;
  flex-direction: column;
  gap: var(--space-md);
}

.form-group {
  display: flex;
  flex-direction: column;
  gap: var(--space-sm);
}

.form-label {
  font-size: 0.875rem;
  color: var(--text-secondary);
  font-weight: 500;
  transition: var(--transition-fast);
}

.form-control {
  padding: var(--space-md);
  border-radius: var(--radius-md);
  border: 2px solid var(--border-color);
  background: linear-gradient(90deg, #ff69b4, #1e90ff, #00ff7f);
  background-size: 200% 200%;
  animation: inputGlow 4s ease-in-out infinite;
  color: var(--text-primary);
  font-family: inherit;
  font-size: var(--font-size-base);
  transition: var(--transition-normal);
  box-shadow: 0 0 10px var(--glow-color);
}

.form-control:focus {

```

```

outline: none;
border-color: var(--accent-primary);
box-shadow: 0 0 20px var(--glow-color), 0 0 0 2px var(--accent-primary);
animation-duration: 1s;
}

@keyframes inputGlow {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}

/* Button styles */
.btn {
  width: 100%;
  padding: var(--space-md);
  border-radius: var(--radius-md);
  border: 2px solid var(--border-color);
  background: linear-gradient(45deg, var(--accent-primary), var(--accent-secondary));
  background-size: 200% 200%;
  animation: buttonPulse 2s ease-in-out infinite;
  color: white;
  font-family: inherit;
  font-size: var(--font-size-base);
  font-weight: 900;
  cursor: pointer;
  transition: var(--transition-normal);
  display: inline-flex;
  justify-content: center;
  align-items: center;
  gap: var(--space-sm);
  text-transform: uppercase;
  letter-spacing: 0.1em;
  box-shadow: 0 0 15px var(--glow-color);
  text-shadow: 0 0 5px var(--text-primary);
}

```

```

.btn: hover {
  animation-duration: 0.5s;
  box-shadow: 0 0 25px var(--glow-color);
  transform: scale(1.05);
}

.btn: active {
  transform: scale(0.95);
}

.btn: focus {
  outline: none;
  box-shadow: 0 0 30px var(--glow-color), 0 0 0 3px var(--accent-primary);
}

@keyframes buttonPulse {
  0% { background-position: 0% 50%; box-shadow: 0 0 15px var(--glow-color); }
  50% { background-position: 100% 50%; box-shadow: 0 0 25px var(--glow-color); }
  100% { background-position: 0% 50%; box-shadow: 0 0 15px var(--glow-color); }
}

/* Response styles */
.response {
  margin-top: var(--space-lg);
  padding: var(--space-md);
  border-radius: var(--radius-md);
  background: var(--bg-tertiary);
  color: var(--text-primary);
  border: 1px solid var(--border-color);
  opacity: 0;
  transform: translateY(-10px);
  transition: var(--transition-normal);
  pointer-events: none;
}

.response.show {
  opacity: 1;
}

```

```

    transform: translateY(0);
    pointer-events: auto;
}

/* Responsive styles */
@media (max-width: 768px) {
    .container {
        padding: var(--space-xl);
    }

    .title {
        font-size: 1.3rem;
    }
}

@media (max-width: 480px) {
    body {
        padding: var(--space-md);
    }

    .container {
        border-radius: var(--radius-lg);
        padding: var(--space-lg);
    }
}

/* Scrollbar styles */
::-webkit-scrollbar {
    width: 10px;
    height: 10px;
}

::-webkit-scrollbar-track {
    background: var(--bg-secondary);
}

::-webkit-scrollbar-thumb {

```

```

    background: var(--accent-primary);
    border-radius: var(--radius-sm);
}

::-webkit-scrollbar-thumb:hover {
    background: var(--accent-secondary);
}

/* Accessibility styles */
:focus-visible {
    outline: 2px solid var(--accent-primary);
    outline-offset: 2px;
}

/* Y2K Decorative elements */
.sparkle {
    position: absolute;
    width: 10px;
    height: 10px;
    background: radial-gradient(circle, var(--text-primary), var(--accent-primary));
    border-radius: 50%;
    animation: sparkleTwinkle 1.5s ease-in-out infinite;
    pointer-events: none;
}

.sparkle:nth-child(1) { top: 10%; left: 10%; animation-delay: 0s; }
.sparkle:nth-child(2) { top: 20%; right: 15%; animation-delay: 0.5s; }
.sparkle:nth-child(3) { bottom: 30%; left: 20%; animation-delay: 1s; }
.sparkle:nth-child(4) { top: 50%; right: 10%; animation-delay: 1.5s; }
.sparkle:nth-child(5) { bottom: 20%; left: 50%; animation-delay: 2s; }

@keyframes sparkleTwinkle {
    0%, 100% { opacity: 0; transform: scale(0.5); }
    50% { opacity: 1; transform: scale(1); }
}

/* Add sparkles to body */

```

```

body::before {
  content: '';
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  pointer-events: none;
  z-index: -1;
}

/* Removed emojis */

/* Spinning Meme Images */
.spinning-meme {
  position: fixed;
  width: 100px;
  height: 100px;
  background-size: cover;
  background-repeat: no-repeat;
  animation: spinAndFloat 8s linear infinite;
  z-index: -1;
  pointer-events: none;
}

.spinning-meme:nth-child(1) {
  top: 10%;
  left: 5%;
  animation-delay: 0s;
}

.spinning-meme:nth-child(2) {
  top: 20%;
  right: 10%;
  animation-delay: 2s;
}

```

```
.spinning-meme:nth-child(3) {  
  bottom: 30%;  
  left: 15%;  
  animation-delay: 4s;  
}
```

```
.spinning-meme:nth-child(4) {  
  top: 60%;  
  right: 5%;  
  animation-delay: 6s;  
}
```

```
.spinning-meme:nth-child(5) {  
  bottom: 10%;  
  left: 60%;  
  animation-delay: 8s;  
}
```

```
.spinning-meme:nth-child(6) {  
  top: 5%;  
  left: 50%;  
  animation-delay: 1s;  
}
```

```
.spinning-meme:nth-child(7) {  
  top: 70%;  
  left: 20%;  
  animation-delay: 3s;  
}
```

```
.spinning-meme:nth-child(8) {  
  bottom: 50%;  
  right: 20%;  
  animation-delay: 5s;  
}
```

```
.spinning-meme:nth-child(9) {
```

```

    top: 40%;
    left: 70%;
    animation-delay: 7s;
}

.spinning-meme:nth-child(10) {
    bottom: 5%;
    right: 50%;
    animation-delay: 9s;
}

.spinning-meme:nth-child(11) {
    top: 30%;
    left: 30%;
    animation-delay: 2s;
}

.spinning-meme:nth-child(12) {
    bottom: 70%;
    right: 30%;
    animation-delay: 4s;
}

.spinning-meme:nth-child(13) {
    top: 80%;
    left: 40%;
    animation-delay: 6s;
}

.spinning-meme:nth-child(14) {
    bottom: 40%;
    left: 80%;
    animation-delay: 8s;
}

.spinning-meme:nth-child(15) {
    top: 15%;

```

```

    right: 60%;
    animation-delay: 10s;
}

@keyframes spinAndFloat {
    0% { transform: translateY(0px) rotate(0deg) scale(1); }
    25% { transform: translateY(-20px) rotate(90deg) scale(1.1); }
    50% { transform: translateY(-40px) rotate(180deg) scale(1); }
    75% { transform: translateY(-20px) rotate(270deg) scale(0.9); }
    100% { transform: translateY(0px) rotate(360deg) scale(1); }
}

```

## Приложение А4. Исходный код server.js

```

const express = require("express");
const path = require("path");
const bodyParser = require("body-parser");
const multer = require("multer");
const fs = require("fs");

const app = express();
const port = 8080;

console.log("[INIT] Starting server...");
console.log("[PATH] __dirname =", __dirname);
console.log("[PATH] static dir =", path.resolve(__dirname, "public"));

// логируем запрос
app.use((req, res, next) => {
    console.log(`[REQ] ${req.method} ${req.url}`);
    next();
});

// лог статик
app.use(
    express.static(path.resolve(__dirname, "public"), {
        setHeaders(res, filePath) {

```

```

        console.log(`[STATIC] Sent file: ${filePath}`);
    },
  })
);

app.use(bodyParser.urlencoded({ extended: true }));

// Ensure uploads directory exists
const uploadsDir = path.join(__dirname, 'public', 'uploads');
if (!fs.existsSync(uploadsDir)) {
  fs.mkdirSync(uploadsDir, { recursive: true });
}

// Multer storage
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, uploadsDir);
  },
  filename: (req, file, cb) => {
    const uniqueName = Date.now() + '-' + Math.round(Math.random() * 1E9) + path.extname(file.originalname);
    cb(null, uniqueName);
  }
});

const upload = multer({ storage });

const list = [];
const images = [];

// имплементация GET /list
app.get("/list", (req, res) => {
  console.log("[GET /list] returning list:", list);
  res.send(list);
});

// имплементация GET /images
app.get("/images", (req, res) => {

```

```

    console.log("[GET /images] returning images:", images);
    res.json(images);
  });

  // возмущаем отдачу index.html
  app.get("/", (req, res) => {
    const file = path.resolve(__dirname, "index.html");
    console.log("[GET /] sending", file);
    res.sendFile(file);
  });

  // возмущаем POST
  app.post("/", upload.single('meme'), (req, res) => {
    console.log("[POST /] BODY =", req.body);
    console.log("[POST /] FILE =", req.file);

    const data = { ...req.body };
    if (req.file) {
      const imagePath = 'uploads/' + req.file.filename;
      data.meme = imagePath;
      images.push(imagePath);
    }

    list.push(data);
    console.log("[POST /] list now =", list);
    console.log("[POST /] images now =", images);
    res.send("OK");
  });

  // заныск сервера
  app.listen(port, () => {
    console.log(`Server running at http://localhost:${port}`);
  });

```