STEP ACADEMY

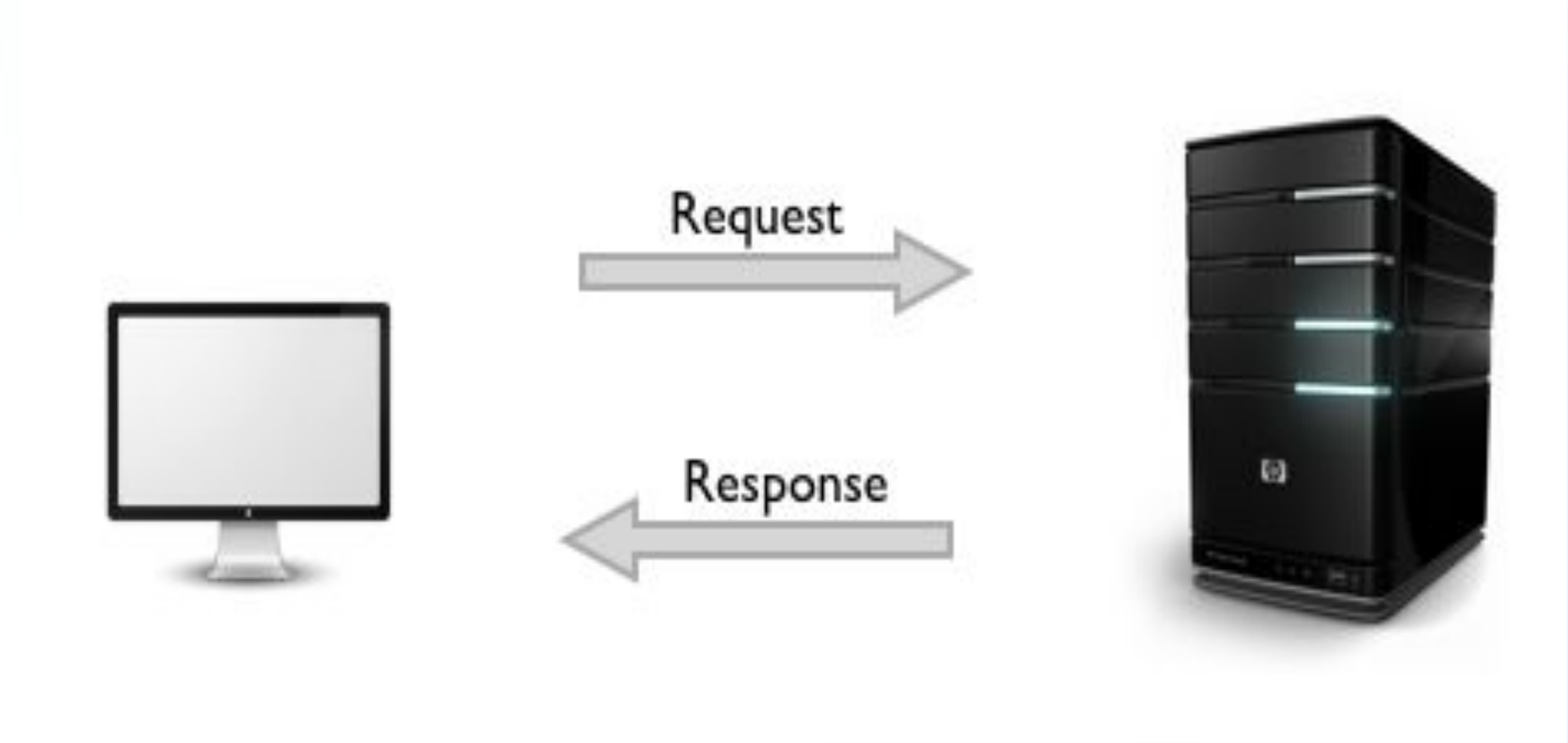
ЗАНЯТИЕ №5

Socket.io

Что такое socket.io

Библиотека node.js, предоставляющая обмен сообщениями между клиентом и сервером в режиме реального времени.

Стандартный обмен сообщениями



socket.io

socket.io позволяет установить прямое соединение, прослушивая порты и ожидая сообщения.
Самый простой пример: чат

Технологии

- WebSocket
- Adobe Flash Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

В данном списке по порядку указаны технологии, которые использует Socket.io

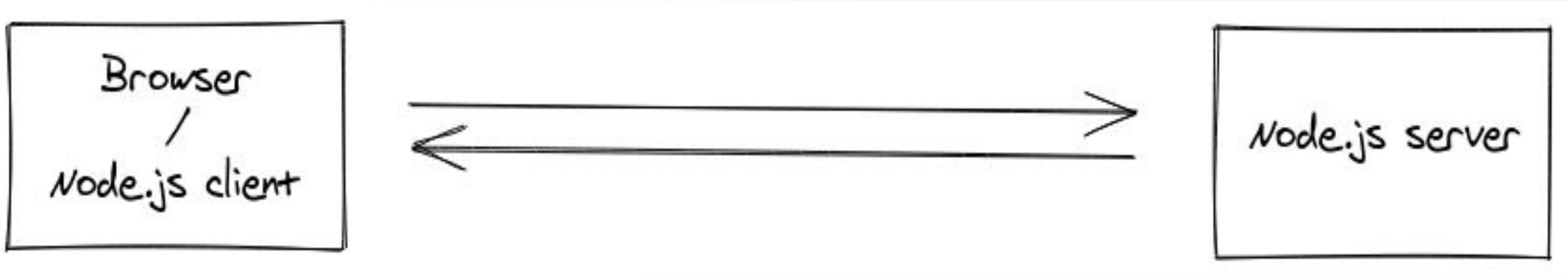
Деление socket.io

Клиентская часть

(браузер или программа)

Серверная часть

Любой сервер



Вариации socket.io

`Socket.io` можно использовать различными способами, существует большое количество `API` под разные языки.

Мы будем рассматривать примеры браузерного использования `socket.io` с сервером на `node.js`

Понятия socket

Сервер (Server) - сервер, обслуживающий соединение

Клиент (Client) - клиент, подключившийся к серверу

Комнаты (Rooms) - комнаты, отделяющие разные подключения

Установка socket.io

npm install socket.io

Настройка сервера

Понадобится express, http и socket.io
http нужен для отдельного подключения socket.io

Подключение библиотек и настройка

```
let express = require('express');
let app = express();
let http = require('http');
let server = http.createServer(app);
let { Server } = require("socket.io");
let io = new Server(server);
```

Код сервера

```
app.get('/', (req, res) => {
    res.sendFile(__dirname + '/index.html');
  });

  io.on('connection', (socket) => {
    console.log('a user connected');
  });

  server.listen(3000, () => {
    console.log('listening on *:3000');
  });
```

io.on

io как экземпляр сервера socket работает на событийной системе node.js, и принимает 2 параметра: имя события и callback функцию.
Мы использовали событие connection, однако не видим результат работы.
Для этого нужно сделать клиентскую часть

Клиентский код

```
<input type="text" id="msg">
<button id="send">Send</button>
<div id="messages"></div>
<script src="/socket.io/socket.io.js"></script>
<script>
      let socket = io();
</script>
```

Запуск и первый результат

При выполнении функции io() срабатывает подключение к серверу, и мы видел результат события connection

Объект socket

Объект socket, который принимает callback-функция - это объект, который может прослушивать и вызывать события, а также имеет некоторые полезные свойства

Отправка события с помощью socket

```
io.on("connection", function(socket){
    console.log("user connected");
    socket.emit("message", {text: "Welcome", chatId:
socket.id});
})
```

“message” - это имя события, выбранное разработчиков, второй параметр - это данные, отправленные клиенту по данному событию.

Второй параметр может быть как объектом, так и простым значением

Сообщение на frontend

```
let socket = io();
        socket.on("message", function(message){
            console.log(message);
        })
```

message - это данные пришедшие от сервера

Отправка сообщения из клиентского кода

```
let msgBox = document.getElementById("messages");
        let msgText = document.getElementById("msg");
        let sendBtn = document.getElementById("send");
        sendBtn.addEventListener("click", function(){
            socket.emit("chatMessage", {text:
msgText.value});
        })
```

emit - метод, который вызывает событие

Обработка входящих сообщений

Frontend

```
let socket = io();
        socket.on("message", function(message){
            let div = document.createElement("div");
            div.textContent = message.text;
            msgBox.append(div);
        })
```

on - метод, который срабатывает при событии “message” сервера

Отправка сообщений сервером

```
socket.emit("message", message);
```

Отправляет сообщение только по данному подключению

```
io.emit("message", message);
```

Отправляет сообщение всем подключениям

```
socket.broadcast.emit("message", message);
```

Отправляет сообщения всем подключениям, кроме текущего

События socket

- connection - подключение
- disconnect - отключение

Подключение к комнате

```
socket.join("room 1");
```

join - метод для подключения к комнате. Комната указывается в виде строки с меткой комнаты. Можно указать несколько комнат в виде массива

По умолчанию каждый socket подключается к уникальной комнате с именем, равным socket.id

Благодаря этому можно отправлять сообщения конкретному пользователю

Отправка сообщений в конкретную комнату

```
io.to("room 1").emit("message", "Message from room")
```

Метод to выберет для отправки сообщения только комнату room 1

```
socket.broadcast.to("room 1").emit("message", message);
```

Выбор комнаты для отправки есть и у socket.broadcast, и у socket

Работа с комнатами

```
socket.rooms
```

Множество (Set) из всех комнат данного подключения

```
socket.leave("room 1")
```

Покинуть указанную комнату

Дополнительные ссылки

Официальный сайт socket.io (документация на английском)

Неполный перевод на русский