УДК 004

Создание онлайн чата с помощью технологии WebSocket на Node.Js

Круглик Роман Игоревич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В статье рассматривается метод создания онлайн чата с помощью технологии WebSocket. Данная технология создаёт интерактивное соединение между клиентом (браузером) и сервером на платформе Node.Js.
Ключевые слова: JavaScript, WebSocket, Node.Js.

Create online chat using WebSocket technology on Node.Js

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article discusses the method of creating online chat using WebSocket technology. This technology creates an interactive connection between a client (browser) and a server on the Node.Js platform.
Keywords: JavaScript, WebSocket, Node.Js.

Веб-сокеты (Web Sockets) — это передовая технология, которая позволяет создавать интерактивное соединение между клиентом (браузером) и сервером для обмена сообщениями в режиме реального времени. Веб-сокеты, в отличие от HTTP, позволяют работать с двунаправленным потоком данных, что делает эту технологию совершенно уникальной.

Веб-сокетам для ответа не нужны повторяющиеся запросы. Достаточно выполнить один запрос и ждать отклика. Веб-сокеты чаще всего используются в:

1. приложениях реального времени;
2. приложениях, где происходит онлайн общение;
3. IoT-приложениях;
4. многопользовательских играх.

Областью применения различных паттернов проектирования интересуются многие. В статье Д.В. Двоеглазов, И.П. Дешко, К.Г.Кряженков, А.А. Тихонов [1] рассказывают о применении технологии websocket в системе удаленного доступа к лабораторным стендам с инфокоммуникационным оборудованием. С.А. Васильев [2] приводит основные преимущества использования протокола передачи данных WebSocket и специфика их достижения для построения веб-сервисов. В

статье И.В. Синицин, Е.А. Леонов, А.В. Аверченков, С.А. Шептунов рассказывают о разработке метода репликации данных между клиентами в многопользовательских веб-приложениях реального времени, построенных на базе протокола websocket.

В данной статье рассматривается разработка чата для общения людей в реальном времени. Так как WebSocket обращается именно к серверу на JavaScript коде, нам понадобится установленный Node.Js. Для начала создадим главный файл с разметкой (см. рис. 1).

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style...>
</head>
<body>
<div class="main-class">
    <div id="text-area"></div>
    <div class="text-area-submit">
        <input type="text" id="text">
        <button>Отправить</button>
    </div>
</div>
<script>
    var socket = new WebSocket("ws://localhost:5001");
    var area = document.getElementById('text-area');
    socket.onmessage = function (event) {
        area.innerHTML +=event.data+"<br>";

    }
    document.querySelector('button').onclick = function () {
        var text = document.getElementById('text').value;
        socket.send(text);
    };
</script>
</body>
</html>
```

Рисунок 1. Файл index.html

В начале идёт разметка и стили. После чего создаётся объект new WebSocket, который выполняет соединение через специальный протокол ws. Если на сервере есть не прочитанное сообщение он принимает и вставляет его в html поле. Функция onclick считывает текстовое поле и отправляет на сервер введённое сообщение. Посмотрим на интерфейс чата (см. рис. 2).

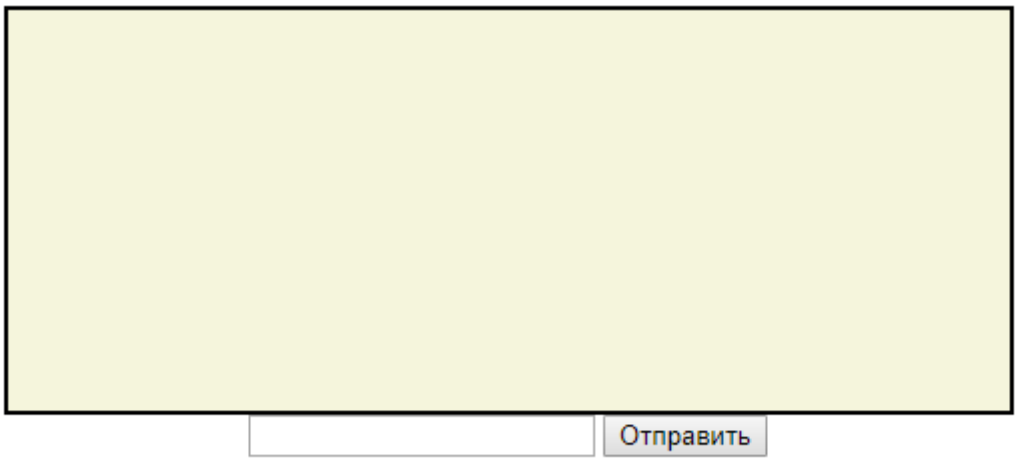

Рисунок 2. Интерфейс онлайн чата

Для того, чтобы чат работал в онлайн режиме, нужно запрограммировать серверную часть нашего чата, которая будет обрабатывать сообщения (см. рис. 3).

```
var server = require('ws').Server;
var s = new server({port:5001});

s.on('connection',function (ws) {
    ws.on('message',function (message) {
        s.clients.forEach(function e(client) {
            client.send(message)
        })
    });
    ws.on('close',function () {
        console.log("Ошибка");
    })
})
```

Рисунок 3. Код серверной части

Смысл сервера заключается в том, что он получает сообщение, которое передаёт ему один пользователь и возвращает всем остальным участникам чата. Для работы скрипта, необходимо активировать сервер Node.Js и создать объект с указанием порта. После чего проведём эксперимент с 3 пользователями, которые присоединились к одному чату. При отправке сообщения с одной вкладки, мы получать его в других, тем самым идёт онлайн общение. При входе нового пользователя в чат он не увидит архив сообщений, так как они не куда не сохраняются.

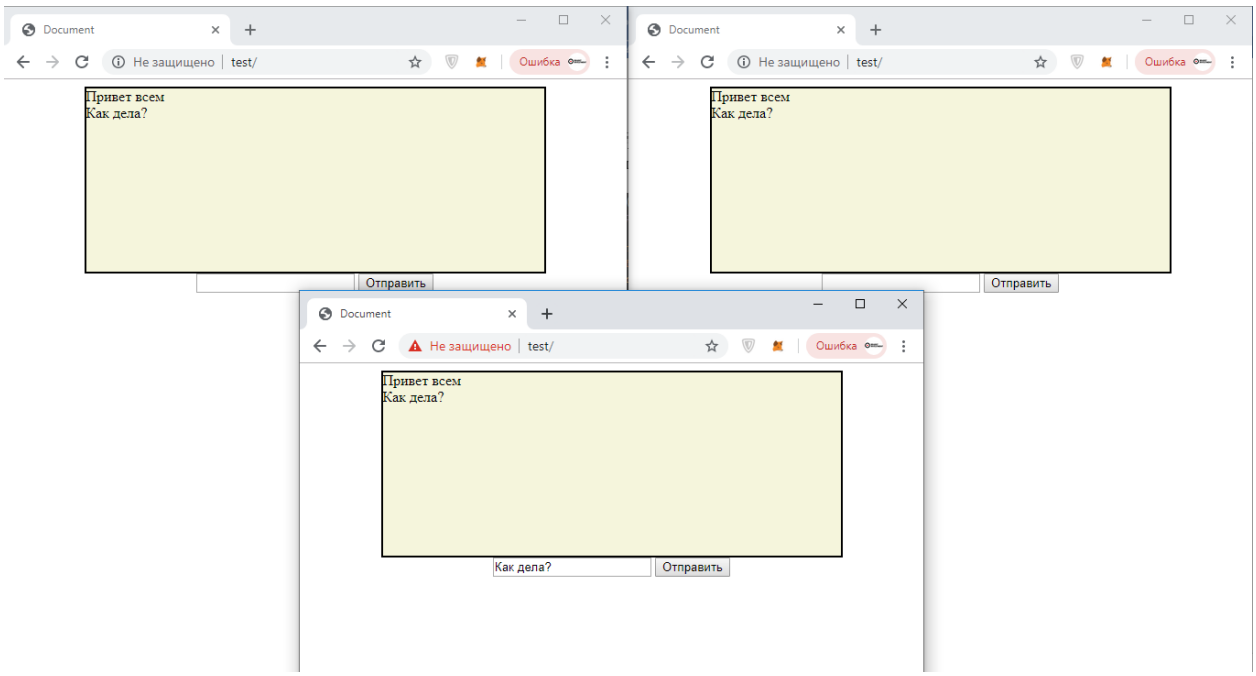

Рисунок 4. Общение между 3 пользователями

В результате во всех вкладках мы видим одинаковые сообщения. Так же мы можем спрашивать при входе в чат пользователя его имя и выводить его вместе с сообщением. Данная работа станет основой для дальнейшего улучшения интерфейса и функционала онлайн чата. Так же с помощью WebSocket можно рисовать графики, передавать изображения и видео.

Библиографический список

1. Двоеглазов Д.В., Дешко И.П., Кряженков К.Г., Тихонов А.А. Применение технологии websocket в системе удаленного доступа к лабораторным стендам с инфокоммуникационным оборудованием // Интернет-журнал Науковедение. 2015. Т. 7. № 4 (29). С. 69.
2. Васильев С.А. Методы построения распределенных сервис-ориентированных электронных образовательных систем на основе протокола websocket // Современное образование: содержание, технологии, качество. 2017. Т. 1. С. 137-139.
3. Синицин И.В., Леонов Е.А., Аверченков А.В., Шептунов С.А. Разработка метода репликации данных между клиентами в многопользовательских веб-приложениях реального времени, построенных на базе протокола websocket // Инфокоммуникационные технологии. 2018. Т. 16. № 4. С. 424-430.