УДК 004.021

ПРОЕКТИРОВАНИЕ REALTIME ВЕБ-ПРИЛОЖЕНИЙ ПРИ ПОМОЩИ WEBSOCKET И СЕРВЕРНЫХ СОБЫТИЙ

А.А. ЛЕВИЦКИЙ
(Представлено: Т.С. РУДЬКОВА)

Представлено теоретическое объяснение работы WebSocket-протокола, а также практический способ создания realtime веб-приложений при помощи данной технологии. Также содержится историческая справка о становлении данной технологии стандартом.

Когда данные на сервере меняются, пользователям необходимо дать об этом знать без необходимости действий с их стороны. Это один из видов увеличения производительности, который избавляет пользователя от однотипных действий, таких как обновление страницы.

Ранее приходилось эмулировать серверные события, используя для этого метод long-polling, заставляющий пользователей делать «длинные» запросы, которые оставались открытыми до того, как сервер будет готов отправить какое-либо сообщение в ответ. После получения сообщения, запрос будет закрыт, а новый – сформирован. Другое решение этой проблемы – <iframe> хаки и Flash, но данные методы не идеальны, так как требуют реализации нестандартного поведения технологий, которые не предназначены для задач, связанных с серверными событиями, а также неудобны в использовании.

Принцип работы WebSocket и алгоритмы его реализации. В 2006 году компания Opera представила серверные события (SSE) из спецификаций WHATWG Web Applications 1.0. Эта технология позволяла потоково передавать события непрерывно из веб-сервера прямо в клиентский браузер. Другие браузеры последовали этому примеру и стали реализовывать этот стандарт в 2011 году как часть спецификаций HTML5. В этом же году WebSocket-протокол был стандартизирован [2].

WebSocket позволяет открыть двустороннее соединение между клиентом и сервером, предоставляя возможность пересылать сообщения клиенту, когда какие-либо данные на сервере меняются, без необходимости клиента запрашивать эти самые данные. Это самая важная вещь для отзывчивости приложения с большим количеством одновременных соединений, например, онлайн-игры или мессенджеры. Однако до появления библиотеки socket.io, вышедшей в 2014 году, это было затруднительно [1]. Впоследствии эта библиотека стала стандартом использования WebSocket в современных веб-приложениях.

Конструктор WebSocket инициализирует соединение с сервером через протокол ws или wss (secure – безопасный). Этот протокол поддерживают все современные браузеры как на компьютерах, так и на смартфонах. Таблица поддержки браузеров проиллюстрирована на рисунке 1.

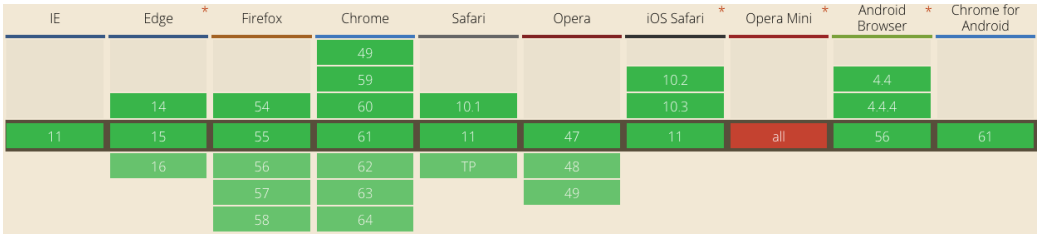

Рисунок 1. – Поддержка браузеров WebSocket

Однако такая технология, как WebSocket, требует безупречной отладки, что невозможно достичь без специальных инструментов. Например, Google Chrome предоставляет возможность наблюдать сообщения при помощи вкладки Frame (рисунок 2).

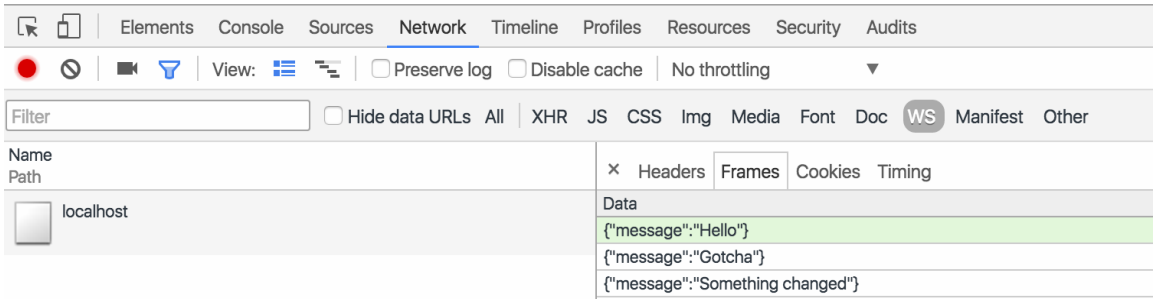

Рисунок. 2. – Отладка WebSocket в Chrome

Реализация WebSocket должна присутствовать как на клиентской стороне (по средствам браузерного JavaScript или других языков, транслирующихся в JavaScript, например, Dart), так и на серверной стороне. Наиболее приспособленный для этого серверный язык – NodeJS, в котором реализация WS является встроенной. Примеры кода сообщений на клиентской и серверной части WebSocket приведены в таблице 1.

Таблица 1. – Пример сообщений клиентской и серверной части WebSocket

Client	Server

let socket = new WebSocket('ws://localhost:8081/');
socket.onopen = function() {
let json = JSON.stringify({ message:'Hello ' });
socket.send(json);
}
socket.onmessage = function(event) {
console.log(event.data);
}
socket.onerror = function(event){
console.log(event);
}
socket.onclose = function(code, reason) {
console.log(code, reason);
}
window.addEventListener('beforeunload', function() {
socket.close();});
	let WSS = require('ws').Server;
	let wss = new WSS({ port: 8081 });
	wss.on('connection', function(socket) {
	console.log('Opened');
	let json = JSON.stringify({ message: 'Gotcha' });
	socket.send(json);
	socket.on('message', function(message) {
	console.log('Received: '); });
	socket.on('close', function() {
	console.log('Closed ');
	});
	});

Заключение

Рассмотренный протокол WebSocket, на основе которого строятся современные realtime веб-приложения, позволяет передавать сообщения в любом количестве без необходимости клиентской части приложения запрашивать эти данные. На основе этой технологии удобно передавать сообщения для чатов, а также другие чувствительные данные по зашифрованному каналу, что не могут конкурирующие технологии. Поэтому для повышения скорости и удобства корпоративной переписки необходимо разработать программный продукт, который будет включать в себя реализацию данной технологии. Данная тема является актуальной, так как комплексного решения этой задачи в настоящее время не существует.

ЛИТЕРАТУРА

1. WebSocket [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/WebSocket. – Дата доступа: 17.09.2017.
2. WebSockets Web API [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. – Дата доступа: 18.09.2017.