УДК 004.582

ИСПОЛЬЗОВАНИЕ SOCKET.IO ДЛЯ АСИНХРОННОГО И СИММЕТРИЧНОГО ОБМЕНА ДАННЫМИ

М.А. БАЛАБАШ
(Представлено: Д.В. ПЯТКИН)

Рассматривается javascript-библиотека с открытым исходным socket.io, при помощи которой можно осуществлять обмен данными между клиентом и сервером в реальном времени.

Интернет создавался главным образом на основе так называемой парадигмы запросов и ответов, реализованной с помощью протокола HTTP. Она заключается в том, что пока клиент не отправит запрос на открытие следующей страницы, она не загрузится. Примерно с 2005 года с появлением технологии AJAX работа в Интернете стала более динамичной. При этом обмен данными по протоколу HTTP все равно инициировался клиентом, и это требовало от пользователя выполнения определенных действий или периодического опроса сервера для загрузки обновленной информации.

Технологии, позволяющие серверу отправлять клиенту новые данные в момент их появления, существуют довольно давно. Они известны как Push или Comet. Один из распространенных способов создания иллюзии соединения, инициируемого сервером. Это так называемый *метод длинного опроса*. Его суть в том, что клиент создает подключение к HTTP-серверу и оно не закрывается до отправки ответа. Если на сервере действительно есть обновленные данные, он отправляет ответ (в других технологиях для этого используются Flash, multipart-запросы XHR и особые HTML-файлы). Технология длинного опроса, как и другие подобные приемы, прекрасно работает. Вы пользуетесь ими каждый день, например, в чате Gmail.

Однако у этого метода есть один недостаток – он использует протокол HTTP, что плохо подходит для приложений с низким временем реакции. В частности, имеем в виду многопользовательские браузерные игры от первого лица и другие сетевые игры в реальном времени.

Основная часть

В спецификации WebSocket определен API для создания сокетных соединений между браузером и сервером. Иными словами, между клиентом и сервером устанавливается постоянное подключение, в котором обе стороны могут инициировать обмен данными.

Socket.IO главным образом использует протокол WebSocket, но если нужно, использует другие методы, например Adobe Flash сокеты, JSONP запросы или AJAX запросы, предоставляя тот же самый интерфейс. Помимо того, что Socket.IO может быть использована как оболочка для WebSocket, она содержит много других функций, включая вещание на несколько сокетов, хранение данных, связанных с каждым клиентом, и асинхронный ввод/вывод.

Основными сферами применения технологии являются:

1. Быстрый обмен данными для онлайн игр, чатов и пр.
2. Веб-приложения с интенсивным обменом данными, требовательные к скорости обмена и каналу.
3. Push уведомления
4. IoT-приложения.
5. Удаленный доступ.

Преимущества данного подхода:

1. Высокая скорость и эффективность передачи – обеспечивает малый размер передаваемых данных, который иногда даже будет помещаться в один TCP-пакет – здесь, конечно, же все зависит от вашей бизнес-логики.

2. В отличие от http, веб-сокеты не имеют ограничений на время жизни в неактивном состоянии. Это значит, что больше не надо периодически рефрешить соединение, так как его не вправе «прихлопывать» всякие прокси. Значит, соединение может висеть в неактивном виде и не требовать ресурсов. Конечно, можно возразить, что на сервере будут забиваться TCP-сокеты. Для этого достаточно использовать хороший мультиплексор, и нормальный сервер легко потянет до миллиона открытых коннектов.

3. Как известно в HTTP предусмотрено ограничение на число одновременных октрытых сессий к одному серверу. Из-за этого если у вас много различных асинхронных блоков на странице, то вам приходилось делать не только серверный, но и клиентский мультиплексор. Однако это ограничение не распространяется на веб-сокеты. Открываете столько, сколько вам нужно. А сколько использовать – одно (и через него все мультиплексировать) или же наоборот – на каждый блок свое соединение – решать вам. Исходите из удобства разработки, нагрузки на сервер и клиент.

4. Возможность разрабатывать кросс-доменные приложения. AJAX имеет ряд ограничений на создание кросс-доменных запросов. WebSockets не имеет таких ограничений.

Socket.io подходит для использования на клиенте и сервере (разработанным с использованием технологии node,js). Для работы с технологией WebSockets на сервере существует множество реализаций (Jetty, EventMachine, Tornado, Shirasu, libwebsockets, SuperWebSocket).

Работа с библиотекой не представляет большой сложности.

Пример создания сервера на node.js:

Листинг 1 – Пример создание Socket.io сервера

```
var io = require('socket.io');
var http = require('http');
var app = http .createServer();
var io = io.listen(app);
app.listen(80);

io.sockets.on('connection', function (socket) {
  socket.on('eventServer', function (data) {
            console.log(data);
            socket.emit('eventClient', { data: 'Hello Client' });
  });
  socket.on('disconnect', function () {
            console.log('user disconnected');
  });
});
```

Схема работы с Socket.io на клиенте не отличается от работы с обычной библиотекой.

Пример создания клиента:

Листинг 2 – Пример создания Socket.io клиента

```
 <html>
<head>
<title>Test socket.io</title>
<script src="https://cdn.socket.io/socket.io-1.2.1.js"></script>
</head>
<body>
<script>
  var socket = io.connect('http://localhost');
  socket.on('eventClient', function (data) {
            console.log(data);
  });
  socket.emit('eventServer', { data: 'Hello Server' });
</script>
</body>
</html>
```

Протоколом проводной связи (установления соединения и обмена данными между клиентом и сервером) для технологии WebSocket является RFC6455. Последние версии Chrome и Chrome для Android полностью соответствуют спецификации RFC6455, в том числе и в части отправки двоичных сообщений. Этот протокол также поддерживают браузеры Firefox 11 и Internet Explorer 10. На текущий момент технологию WebSockets поддерживают 93% браузеров [4].

Согласно последней версии спецификации, объект WebSocket также способен принимать двоичные сообщения. Форматом по умолчанию является blob.

Socket.io имеет множество методов рассылки сообщений.

Рассмотрим некоторые из них:

Листинг 3 – Пример нескольких способов рассылки сообщений

```
// отправить текущему сокету сформировавшему запрос (туда откуда пришла)
socket.emit('eventClient', "this is a test");

// отправить всем пользователям, включая отправителя
 io.sockets.emit ('eventClient', "this is a test");
```

```
// отправить всем, кроме отправителя
socket.broadcast.emit('eventClient', "this is a test");

// отправить всем клиентам в комнате (канале) 'game', кроме отправителя
socket.broadcast.to('game').emit('eventClient', 'nice game');

// отправить всем клиентам в комнате (канале) 'game', включая отправителя
io.sockets.in('game').emit('eventClient', 'cool game');

// отправить конкретному сокету, по socketid
io.sockets.socket(socketid).emit('eventClient', 'for your eyes only').
```

Использование протокола WebSocket меняет подход к применению серверных приложений. Хотя традиционные наборы серверного программного обеспечения, например LAMP, разработаны для обычного цикла запросов и ответов на базе HTTP, они часто плохо работают в условиях большого числа WebSocket-подключений.

Большое количество одновременных соединений требует принципиально новой архитектуры, которая сочетает низкую ресурсоемкость с возможностями параллельной работы. Такие архитектуры обычно создаются на основе потоковой или так называемой неблокирующей системы обмена данными.

Заключение

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.

ЛИТЕРАТУРА

1. WebSocket [Электронный ресурс] // Wikipedia. – Режим доступа: https://ru.wikipedia.org/wiki/WebSocket. – Дата доступа: 26.09.17.
2. Асинхронный веб, или Что такое веб-сокеты [Электронный ресурс] / Tproger. – Режим доступа: https://tproger.ru/translations/what-are-web-sockets/. – Дата доступа: 26.09.17.
3. Использование WebSocket [Электронный ресурс] / learn.javascript. – Режим доступа: https://learn.javascript.ru/websockets. – Дата доступа: 26.09.17.
4. WebSockets [Электронный ресурс] / caniuse. – Режим доступа: http://caniuse.com/#feat=websockets. – Дата доступа: 26.09.17.