link: null
title: 珠峰架构师成长计划
description: Socket.IO是一个WebSocket库，包括了客户端的js和服务器端的nodejs，它的目标是构建可以在不同浏览器和移动设备上使用的实时应用。
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=120 sentences=294, words=1895

1. socket.io

Socket.IO是一个WebSocket库，包括了客户端的js和服务器端的nodejs，它的目标是构建可以在不同浏览器和移动设备上使用的实时应用。

2. socket.io的特点

- 易用性：socket.io封装了服务端和客户端，使用起来非常简单方便。
- 跨平台：socket.io支持跨平台，这就意味着你有了更多的选择，可以在自己喜欢的平台下开发实时应用。
- 自适应：它会自动根据浏览器从WebSocket、AJAX长轮询、Iframe流等等各种方式中选择最佳的方式来实现网络实时应用，非常方便和人性化，而且支持的浏览器最低达IE5.5。

3. 初步使用

使用npm安装socket.io

```
$ npm install socket.io
```

创建 `app.js` 文件

```
var express = require('express');
var path = require('path');
var app = express();

app.get('/', function (req, res) {
    res.sendFile(path.resolve('index.html'));
});

var server = require('http').createServer(app);
var io = require('socket.io')(server);

io.on('connection', function (socket) {
    console.log('客户端已经连接');
    socket.on('message', function (msg) {
        console.log(msg);
        socket.send('sever:' + msg);
    });
});
server.listen(80);
```

服务端运行后会在根目录动态生成socket.io的客户端js文件 客户端可以通过固定路径 `/socket.io/socket.io.js`添加引用
客户端加载socket.io文件后会得到一个全局的对象io<br> `connect`函数可以接受一个 `url`参数，url可以socket服务的http完整地址，也可以是相对路径，如果省略则表示默认连接当前路径

创建index.html文件

```
<script src="/socket.io/socket.io.js">script>
<script>
 window.onload = function(){
    const socket = io.connect('/');

    socket.on('connect',function(){
        console.log('连接成功');
    });

    socket.on('disconnect',function(){
       console.log('断开连接');
    });
 };
</script>
```

成功建立连接后，我们可以通过 `socket`对象的 `send`函数来互相发送消息 修改index.html

```
var socket = io.connect('/');
socket.on('connect',function(){

   socket.send('welcome');
});

socket.on('message',function(message){
   console.log(message);
});
```

修改app.js

```
var io = require('scoket.io')(server);
io.on('connection',function(socket){

  socket.send('欢迎光临');

  socket.on('message',function(data){
      console.log(data);
  });
});
```

4. 深入分析

- `send`函数只是 `emit`的封装
- `node_modules\socket.io\lib\socket.js`源码

```
function send(){
  var args = toArray(arguments);
  args.unshift('message');
  this.emit.apply(this, args);
  return this;
}
```

`emit`函数有两个参数

- 第一个参数是自定义的事件名称,发送方发送什么类型的事件名称,接收方就可以通过对应的事件名称来监听接收
- 第二个参数是要发送的数据

事件名称 含义 connect 成功连接到服务器 message 接收到服务器发送的消息 disconnect 客户端断开连接 error 监听错误

5. 划分命名空间

- 可以把服务分成多个命名空间，默认/,不同空间内不能通信 ````js

io.on('connection', function (socket) { //向客户端发送消息 socket.send('/ 欢迎光临'); //接收到客户端发过来的消息时触发 socket.on('message',function(data){ console.log('/'+data); }); }); io.of('/news').on('connection', function (socket) { //向客户端发送消息 socket.send('/news 欢迎光临'); //接收到客户端发过来的消息时触发 socket.on('message',function(data){ console.log('/news '+data); }); });

~~~
### 5.2 &#x5BA2;&#x6237;&#x7AEF;&#x8FDE;&#x63A5;&#x547D;&#x540D;&#x7A7A;&#x95F4;
```js
window.onload = function(){
var socket = io.connect('/');
//&#x76D1;&#x542C;&#x4E0E;&#x670D;&#x52A1;&#x5668;&#x7AEF;&#x7684;&#x8FDE;&#x63A5;&#x6210;&#x529F;&#x4E8B;&#x4EF6;
socket.on('connect',function(){
    console.log('&#x8FDE;&#x63A5;&#x6210;&#x529F;');
    socket.send('welcome');
});
socket.on('message',function(message){
    console.log(message);
});
//&#x76D1;&#x542C;&#x4E0E;&#x670D;&#x52A1;&#x5668;&#x7AEF;&#x65AD;&#x5F00;&#x8FDE;&#x63A5;&#x4E8B;&#x4EF6;
socket.on('disconnect',function(){
     console.log('&#x65AD;&#x5F00;&#x8FDE;&#x63A5;');
});

var news_socket = io.connect('/news');
//&#x76D1;&#x542C;&#x4E0E;&#x670D;&#x52A1;&#x5668;&#x7AEF;&#x7684;&#x8FDE;&#x63A5;&#x6210;&#x529F;&#x4E8B;&#x4EF6;
news_socket.on('connect',function(){
    console.log('&#x8FDE;&#x63A5;&#x6210;&#x529F;');
     socket.send('welcome');
});
news_socket.on('message',function(message){
    console.log(message);
});
//&#x76D1;&#x542C;&#x4E0E;&#x670D;&#x52A1;&#x5668;&#x7AEF;&#x65AD;&#x5F00;&#x8FDE;&#x63A5;&#x4E8B;&#x4EF6;
 news_socket.on('disconnect',function(){
    console.log('&#x65AD;&#x5F00;&#x8FDE;&#x63A5;');
});
};
~~~

6. 房间

- 可以把一个命名空间分成多个房间，一个客户端可以同时进入多个房间。
- 如果在大厅里广播 ，那么所有在大厅里的客户端和任何房间内的客户端都能收到消息。
- 所有在房间里的广播和通信都不会影响到房间以外的客户端

```
socket.join('chat');
```

```
socket.leave('chat');
```

7. 全局广播

广播就是向多个客户端都发送消息

```
io.emit('message','全局广播');
```

```
socket.broadcast.emit('message', msg);
socket.broadcast.emit('message', msg);
```

8. 房间内广播

从服务器的角度来提交事件,提交者会包含在内

```
io.in('myroom').emit('message', msg);
io.of('/news').in('myRoom').emit('message',msg);
```

从客户端的角度来提交事件,提交者会排除在外

```
socket.broadcast.to('myroom').emit('message', msg);
socket.broadcast.to('myroom').emit('message', msg);
```

```
io.sockets.adapter.rooms
```

取得进入房间内所对应的所有sockets的hash值，它便是拿到的 `socket.id`

```
 let roomSockets = io.sockets.adapter.rooms[room].sockets;
```

9. 聊天室

- 创建客户端与服务端的websocket通信连接
- 客户端与服务端相互发送消息
- 添加用户名
- 添加私聊
- 进入/离开房间聊天
- 历史消息

app.js

```
let express = require('express');
const path = require('path');
let app = express();
app.get('/news', function (req, res) {
    res.sendFile(path.resolve(__dirname, 'public/news.html'));
});
app.get('/goods', function (req, res) {
    res.sendFile(path.resolve(__dirname, 'public/goods.html'));
});
let server = require('http').createServer(app);
let io = require('socket.io')(server);

let sockets = {};
io.on('connection', function (socket) {

    let rooms = [];
    let username;

    socket.on('message', function (message) {
        if (username) {

            if (rooms.length > 0) {
                for (let i = 0; i < rooms.length; i++) {

                    let result = message.match(/@([^ ]+) (.+)/);
                    if (result) {
                        let toUser = result[1];
                        let content = result[2];
                        sockets[toUser].send({
                            username,
                            content,
                            createAt: new Date()
                        });
                    } else {
                        io.in(rooms[i]).emit('message', {
                            username,
                            content: message,
                            createAt: new Date()
                        });
                    }

                }
            } else {

                let result = message.match(/@([^ ]+) (.+)/);
                if (result) {
                    let toUser = result[1];
                    let content = result[2];
                    sockets[toUser].send({
                        username,
                        content,
                        createAt: new Date()
                    });
                } else {
                    io.emit('message', {
                        username,
                        content: message,
                        createAt: new Date()
                    });
                }
            }
        } else {

            username = message;

            sockets[username] = socket;
            socket.broadcast.emit('message', {
                username: '系统',
                content: `${username} 加入了聊天`,
                createAt: new Date()
            });
        }

    });

    socket.on('join', function (roomName) {
        let oldIndex = rooms.indexOf(roomName);
        if (oldIndex == -1) {
            socket.join(roomName);
            rooms.push(roomName);
        }
    })

    socket.on('leave', function (roomName) {
        let oldIndex = rooms.indexOf(roomName);
        if (oldIndex != -1) {
            socket.leave(roomName);
            rooms.splice(oldIndex, 1);
        }
    });
    socket.on('getRoomInfo', function () {
        console.log(io);

        console.log(io);
    });
});

server.listen(8080);
```

index.html

```html
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://cdn.bootcss.com/bootstrap/3.3.1/css/bootstrap.css">
    <style>
        .user {
            color: green;
            cursor: pointer;
        }
    style>
    <title>聊天室title>
head>

<body>
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2">
                <div class="panel panel-default">
                    <div class="panel-heading text-center">
                        <div>
                            <button class="btn btn-danger" onclick="join('red')">进入红房间button>
                            <button class="btn btn-danger" onclick="leave('red')">离开红房间button>
                        div>
                        <div>
                            <button class="btn btn-success" onclick="join('green')">进入绿房间button>
                            <button class="btn btn-success" onclick="leave('green')">进入绿房间button>
                        div>
                        <div>
                            <button class="btn btn-primary" onclick="getRoomInfo()">
                                获取房间信息
                            button>
                        div>
                    div>
                    <div class="panel-body">
                        <ul class="list-group" id="messages" onclick="clickUser(event)">

                        ul>
                    div>
                    <div class="panel-footer">
                        <div class="row">
                            <div class="col-md-10">
                                <input id="textMsg" type="text" class="form-control">
                            div>
                            <div class="col-md-2">
                                <button type="button" onclick="send()" class="btn btn-primary">发言button>
                            div>
                        div>

                    div>
                div>
            div>
        div>
    div>

    <script src="/socket.io/socket.io.js">script>
    <script>
        let socket = io('/');
        let textMsg = document.querySelector('#textMsg');
        let messagesEle = document.querySelector('#messages');
        socket.on('connect', function () {
            console.log('客户端连接成功');
        });
        socket.on('message', function (messageObj) {
            let li = document.createElement('li');
            li.innerHTML = `${messageObj.username}:${messageObj.content} ${messageObj.createAt.toLocaleString()}`;
            li.className = 'list-group-item';
            messagesEle.appendChild(li);
        });

        function send() {
            let content = textMsg.value;
            if (!content)
                return alert('请输入聊天内容');
            socket.send(content);
        }
        function join(name) {

            socket.emit('join', name);
        }
        function leave(name) {

            socket.emit('leave', name);
        }
        function getRoomInfo() {
            socket.emit('getRoomInfo');
        }
        function clickUser(event) {
            console.log('clickUser', event.target.className);
            if (event.target.className == 'user') {
                let username = event.target.innerHTML;
                textMsg.value = `@${username} `;
            }
        }
    script>
body>

html>
```

10. 聊天室

```javascript
let express = require('express');
let http = require('http');
let path = require('path')
let app = express();
let mysql = require('mysql');
var connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'root',
    database: 'chat'
});
connection.connect();
app.use(express.static(__dirname));
app.get('/', function (req, res) {
    res.header('Content-Type', "text/html;charset=utf8");
    res.sendFile(path.resolve('index.html'));
});

let server = http.createServer(app);

let io = require('socket.io')(server);
const SYSTEM = '系统';

let sockets = {};
let mysockets = {};
let messages = [];

io.on('connection', function (socket) {
    console.log('socket', socket.id)
    mysockets[socket.id] = socket;

    let username;

    let rooms = [];

    socket.on('message', function (message) {
        if (username) {

            let result = message.match(/@([^ ]+) (.+)/);
            if (result) {
                let toUser = result[1];
                let content = result[2];
                let toSocket = sockets[toUser];
                if (toSocket) {
                    toSocket.send({
                        user: username,
                        content,
                        createAt: new Date()
                    });
                } else {
                    socket.send({
                        user: SYSTEM,
                        content: `你私聊的用户不在线`,
                        createAt: new Date()
                    });
                }
            } else {

                let messageObj = {
                    user: username,
                    content: message,
                    createAt: new Date()
                };

                connection.query(`INSERT INTO message(user,content,createAt) VALUES(?,?,?)`, [messageObj.user, messageObj.content, messageObj.createAt],
function (err, results) {
                    console.log(results);
                });
                if (rooms.length > 0) {

                    let targetSockets = {};
                    rooms.forEach(room => {
                        let roomSockets = io.sockets.adapter.rooms[room].sockets;
                        console.log('roomSockets', roomSockets);
                        Object.keys(roomSockets).forEach(socketId => {
                            if (!targetSockets[socketId]) {
                                targetSockets[socketId] = true;
                            }
                        });
                    });
                    Object.keys(targetSockets).forEach(socketId => {
                        mysockets[socketId].emit('message', messageObj);
                    });
                } else {
                    io.emit('message', messageObj);
                }
            }
        } else {

            username = message;

            sockets[username] = socket;

            socket.broadcast.emit('message', { user: SYSTEM, content: `${username}加入了聊天室`, createAt: new Date() });
        }
    });
    socket.on('join', function (roomName) {
        if (rooms.indexOf(roomName) == -1) {

            socket.join(roomName);
            rooms.push(roomName);
            socket.send({
                user: SYSTEM,
                content: `你成功进入了${roomName}房间!`,
                createAt: new Date()
            });

            socket.emit('joined', roomName);
        } else {
            socket.send({
                user: SYSTEM,
                content: `你已经在${roomName}房间了!请不要重复进入!`,
                createAt: new Date()
            });
        }
    });
    socket.on('leave', function (roomName) {
        let index = rooms.indexOf(roomName);
        if (index == -1) {
            socket.send({
                user: SYSTEM,
                content: `你并不在${roomName}房间，离开个毛!`,
                createAt: new Date()
            });
        } else {
            socket.leave(roomName);
            rooms.splice(index, 1);
            socket.send({
                user: SYSTEM,
                content: `你已经离开了${roomName}房间!`,
                createAt: new Date()
            });
            socket.emit('leaved', roomName);
        }
    });
    socket.on('getAllMessages', function () {

        connection.query(`SELECT * FROM message ORDER BY id DESC limit 20`, function (err, results) {

            socket.emit('allMessages', results.reverse());
        });

    });
});
server.listen(8080);
```

```
        .user {
            color: red;
            cursor: pointer;
        }

    socket.io

                        欢迎来到珠峰聊天室

                                进入红房间
                                离开红房间

                                进入绿房间
                                离开绿房间

                                发言

        let contentInput = document.getElementById('content');//输入框
        let messagesUl = document.getElementById('messages');//列表
        let socket = io('/');//io new Websocket();
        socket.on('connect', function () {
            console.log('客户端连接成功');
            //告诉服务器，我是一个新的客户，请给我最近的20条消息
            socket.emit('getAllMessages');
        });
        socket.on('allMessages', function (messages) {
            let html = messages.map(messageObj => `
                <li class="list-group-item"><span class="user">${messageObj.user}</span>:${messageObj.content} <span class="pull-right">${new
Date(messageObj.createAt).toLocaleString()}</span></li>
            `).join('');
            messagesUl.innerHTML = html;
            messagesUl.scrollTop = messagesUl.scrollHeight;
        });
        socket.on('message', function (messageObj) {
            let li = document.createElement('li');
            li.className = "list-group-item";
            li.innerHTML = `<span class="user">${messageObj.user}</span>:${messageObj.content} <span class="pull-right">${new
Date(messageObj.createAt).toLocaleString()}</span>`;
            messagesUl.appendChild(li);
            messagesUl.scrollTop = messagesUl.scrollHeight;
        });

        // click delegate
        function talkTo(event) {
            if (event.target.className == 'user') {
                let username = event.target.innerText;
                contentInput.value = `@${username} `;
            }
        }
        //进入某个房间
        function join(roomName) {
            //告诉服务器，我这个客户端将要在服务器进入某个房间
            socket.emit('join', roomName);
        }
        socket.on('joined', function (roomName) {
            document.querySelector(`#leave-${roomName}`).style.display = 'inline-block';
            document.querySelector(`#join-${roomName}`).style.display = 'none';
        });
        socket.on('leaved', function (roomName) {
            document.querySelector(`#join-${roomName}`).style.display = 'inline-block';
            document.querySelector(`#leave-${roomName}`).style.display = 'none';
        });
        //离开某个房间
        function leave(roomName) {
            socket.emit('leave', roomName);
        }
        function send() {
            let content = contentInput.value;
            if (content) {
                socket.send(content);
                contentInput.value = '';
            } else {
                alert('聊天信息不能为空!');
            }
        }
        function onKey(event) {
            let code = event.keyCode;
            if (code == 13) {
                send();
            }
        }
```

10. 参考