Web领域的实时推送技术，也被称作Realtime技术。这种技术要达到的目的是让用户不需要刷新浏览器就可以获得实时更新。它有着广泛的应用场景，比如在线聊天室、在线客服系统、评论系统、WebIM等。



# WebSocket简介

谈到Web实时推送，就不得不说WebSocket。在WebSocket出现之前，很多网站为了实现实时推送技术，通常采用的方案是轮询 (Polling)和Comet技术，Comet又可细分为两种实现方式，一种是长轮询机制，一种称为流技术，这两种方式实际上是对轮询技术的改进，这些 方案带来很明显的缺点，需要由浏览器对服务器发出HTTP request，大量消耗服务器带宽和资源。面对这种状况，HTML5定义了WebSocket协议，能更好的节省服务器资源和带宽并实现真正意义上的实 时推送。

WebSocket协议本质上是一个基于TCP的协议，它由通信协议和编程API组成，WebSocket能够在浏览器和服务器之间建立双向连接， 以基于事件的方式，赋予浏览器实时通信能力。既然是双向通信，就意味着服务器端和客户端可以同时发送并响应请求，而不再像HTTP的请求和响应。

为了建立一个WebSocket连接，客户端浏览器首先要向服务器发起一个HTTP请求，这个请求和通常的HTTP请求不同，包含了一些附加头信 息，其中附加头信息”Upgrade: WebSocket”表明这是一个申请协议升级的HTTP请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连 接。

一个典型WebSocket客户端请求头：



前面讲到WebSocket是HTML5中新增的一种通信协议，这意味着一部分老版本浏览器（主要是IE10以下版本）并不具备这个功能， [通过百度统计的公开数据显示](http://tongji.baidu.com/data/browser" \t "http://www.open-open.com/lib/view/_blank)，IE8 目前仍以33%的市场份额占据榜首，好在chrome浏览器市场份额逐年上升，现在以超过26%的市场份额位居第二，同时微软前不久宣布停止对IE6的技 术支持并提示用户更新到新版本浏览器，这个曾经让无数前端工程师为之头疼的浏览器有望退出历史舞台，再加上几乎所有的智能手机浏览器都支持HTML5，所 以使得WebSocket的实战意义大增，但是无论如何，我们实际的项目中，仍然要考虑低版本浏览器的兼容方案：在支持WebSocket的浏览器中采用 新技术，而在不支持WebSocket的浏览器里启用Comet来接收发送消息。

# WebSocket实战

本文将以多人在线聊天应用作为实例场景，我们先来确定这个聊天应用的基本需求。

## 需求分析

1、兼容不支持WebSocket的低版本浏览器。  
2、允许客户端有相同的用户名。  
3、进入聊天室后可以看到当前在线的用户和在线人数。  
4、用户上线或退出，所有在线的客户端应该实时更新。  
5、用户发送消息，所有客户端实时收取。

在实际的开发过程中，为了使用WebSocket接口构建Web应用，我们首先需要构建一个实现了 WebSocket规范的服务端，服务端的实现不受平台和开发语言的限制，只需要遵从WebSocket规范即可，目前已经出现了一些比较成熟的 WebSocket服务端实现，比如本文使用的Node.js+Socket.IO。为什么选用这个方案呢？先来简单介绍下他们两。

[回到顶部](http://www.open-open.com/lib/view/open1402479198587.html" \l "_labelTop)

## Node.js

Node.js采用C++语言编写而成，它不是Javascript应用，而是一个Javascript的运行环境，据Node.js创始人 Ryan Dahl回忆，他最初希望采用Ruby来写Node.js，但是后来发现Ruby虚拟机的性能不能满足他的要求，后来他尝试采用V8引擎，所以选择了 C++语言。

Node.js支持的系统包括\*nux、Windows，这意味着程序员可以编写系统级或者服务器端的Javascript代码，交给 Node.js来解释执行。Node.js的Web开发框架Express，可以帮助程序员快速建立web站点，从2009年诞生至今，Node.js的 成长的速度有目共睹，其发展前景获得了技术社区的充分肯定。

[回到顶部](http://www.open-open.com/lib/view/open1402479198587.html" \l "_labelTop)

## Socket.IO

Socket.IO是一个开源的WebSocket库，它通过Node.js实现WebSocket服务端，同时也提供客户端JS库。Socket.IO支持以事件为基础的实时双向通讯，它可以工作在任何平台、浏览器或移动设备。

Socket.IO支持4种协议：WebSocket、htmlfile、xhr-polling、jsonp-polling，它会自动根据浏览 器选择适合的通讯方式，从而让开发者可以聚焦到功能的实现而不是平台的兼容性，同时Socket.IO具有不错的稳定性和性能。

# 编码实现

本文一开始的的插图就是效果演示图：可以[点击这里](http://demo.plhwin.com/chat/" \t "http://www.open-open.com/lib/view/_blank)查看在线演示，整个开发过程非常简单，下面简单记录了开发步骤：

[回到顶部](http://www.open-open.com/lib/view/open1402479198587.html" \l "_labelTop)

## 安装Node.js

根据自己的操作系统，去[Node.js官网](http://www.nodejs.org/download/" \t "http://www.open-open.com/lib/view/_blank)下载安装即可。如果成功安装。在命令行输入node -v和npm -v应该能看到相应的版本号。

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2  3  4 | node -v  v0.10.26  npm -v  1.4.6 |

[回到顶部](http://www.open-open.com/lib/view/open1402479198587.html" \l "_labelTop)

## 搭建WebSocket服务端

这个环节我们尽可能的考虑真实生产环境，把WebSocket后端服务搭建成一个线上可以用域名访问的服务，如果你是在本地开发环境，可以换成本地ip地址，或者使用一个虚拟域名指向本地ip。

先进入到你的工作目录，比如 /workspace/wwwroot/plhwin/realtime.plhwin.com，新建一个名为 package.json的文件，内容如下：

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | {    "name": "realtime-server",    "version": "0.0.1",    "description": "my first realtime server",    "dependencies": {}  } |

接下来使用npm命令安装express和socket.io

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2 | npm install --save express  npm install --save socket.io |

安装成功后，应该可以看到工作目录下生成了一个名为node\_modules的文件夹，里面分别是express和socket.io，接下来可以开始编写服务端的代码了，新建一个文件：index.js

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | var app = require('express')();  var http = require('http').Server(app);  var io = require('socket.io')(http);    app.get('/', function(req, res){      res.send('<h1>Welcome Realtime Server</h1>');  });    http.listen(3000, function(){      console.log('listening on \*:3000');  }); |

命令行运行node index.js，如果一切顺利，你应该会看到返回的listening on \*:3000字样，这说明服务已经成功搭建了。此时浏览器中打开http://localhost:3000应该可以看到正常的欢迎页面。

如果你想要让服务运行在线上服务器，并且可以通过域名访问的话，可以使用Nginx做代理，在nginx.conf中添加如下配置，然后将域名（比如：realtime.plhwin.com）解析到服务器IP即可。

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | server   {     listen       80;     server\_name  realtime.plhwin.com;     location / {       proxy\_pass http://127.0.0.1:3000;     }   } |

完成以上步骤，http://realtime.plhwin.com:3000的后端服务就正常搭建了。



[回到顶部](http://www.open-open.com/lib/view/open1402479198587.html" \l "_labelTop)

## 服务端代码实现

前面讲到的index.js运行在服务端，之前的代码只是一个简单的WebServer欢迎内容，让我们把WebSocket服务端完整的实现代码加入进去，整个服务端就可以处理客户端的请求了。完整的index.js代码如下：

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63 | var app = require('express')();  var http = require('http').Server(app);  var io = require('socket.io')(http);    app.get('/', function(req, res){      res.send('<h1>Welcome Realtime Server</h1>');  });    //在线用户  var onlineUsers = {};  //当前在线人数  var onlineCount = 0;    io.on('connection', function(socket){      console.log('a user connected');        //监听新用户加入      socket.on('login', function(obj){          //将新加入用户的唯一标识当作socket的名称，后面退出的时候会用到          socket.name = obj.userid;            //检查在线列表，如果不在里面就加入          if(!onlineUsers.hasOwnProperty(obj.userid)) {              onlineUsers[obj.userid] = obj.username;              //在线人数+1              onlineCount++;          }            //向所有客户端广播用户加入          io.emit('login', {onlineUsers:onlineUsers, onlineCount:onlineCount, user:obj});          console.log(obj.username+'加入了聊天室');      });        //监听用户退出      socket.on('disconnect', function(){          //将退出的用户从在线列表中删除          if(onlineUsers.hasOwnProperty(socket.name)) {              //退出用户的信息              var obj = {userid:socket.name, username:onlineUsers[socket.name]};                //删除              delete onlineUsers[socket.name];              //在线人数-1              onlineCount--;                //向所有客户端广播用户退出              io.emit('logout', {onlineUsers:onlineUsers, onlineCount:onlineCount, user:obj});              console.log(obj.username+'退出了聊天室');          }      });        //监听用户发布聊天内容      socket.on('message', function(obj){          //向所有客户端广播发布的消息          io.emit('message', obj);          console.log(obj.username+'说：'+obj.content);      });    });    http.listen(3000, function(){      console.log('listening on \*:3000');  }); |

[回到顶部](http://www.open-open.com/lib/view/open1402479198587.html" \l "_labelTop)

## 客户端代码实现

进入客户端工作目录/workspace/wwwroot/plhwin/demo.plhwin.com/chat，新建一个index.html:

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | <!DOCTYPE html>  <html>      <head>          <meta charset="utf-8">          <meta name="format-detection" content="telephone=no"/>          <meta name="format-detection" content="email=no"/>  <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=0" name="viewport">          <title>多人聊天室</title>          <link rel="stylesheet" type="text/css" href="./style.css" />          <!--[if lt IE 8]><script src="./json3.min.js"></script><![endif]-->          <script src="http://realtime.plhwin.com:3000/socket.io/socket.io.js"></script>      </head>      <body>          <div id="loginbox">              <div style="width:260px;margin:200px auto;">                  请先输入你在聊天室的昵称                  <br/>                  <br/>                  <input type="text" style="width:180px;" placeholder="请输入用户名" id="username" name="username" />                  <input type="button" style="width:50px;" value="提交" onclick="CHAT.usernameSubmit();"/>              </div>          </div>          <div id="chatbox" style="display:none;">              <div style="background:#3d3d3d;height: 28px; width: 100%;font-size:12px;">                  <div style="line-height: 28px;color:#fff;">                      <span style="text-align:left;margin-left:10px;">Websocket多人聊天室</span>                      <span style="float:right; margin-right:10px;"><span id="showusername"></span> |                      <a href="javascript:;" onclick="CHAT.logout()" style="color:#fff;">退出</a></span>                  </div>              </div>              <div id="doc">                  <div id="chat">                      <div id="message" class="message">  <div id="onlinecount" style="background:#EFEFF4; font-size:12px; margin-top:10px; margin-left:10px; color:#666;">  </div>                      </div>                      <div class="input-box">                          <div class="input">  <input type="text" maxlength="140" placeholder="请输入聊天内容，按Ctrl提交" id="content" name="content">                          </div>                          <div class="action">                              <button type="button" id="mjr\_send" onclick="CHAT.submit();">提交</button>                          </div>                      </div>                  </div>              </div>          </div>          <script type="text/javascript" src="./client.js"></script>      </body>  </html> |

上面的html内容本身没有什么好说的，我们主要看看里面的4个文件请求：  
1、realtime.plhwin.com:3000/socket.io/socket.io.js  
2、style.css  
3、json3.min.js  
4、client.js

第1个JS是Socket.IO提供的客户端JS文件，在前面安装服务端的步骤中，当npm安装完socket.io并搭建起WebServer后，这个JS文件就可以正常访问了。

第2个style.css文件没什么好说的，就是样式文件而已。

第3个JS只在IE8以下版本的IE浏览器中加载，目的是让这些低版本的IE浏览器也能处理json，这是一个开源的JS，详见：[http://bestiejs.github.io/json3/](http://bestiejs.github.io/json3/" \t "http://www.open-open.com/lib/view/_blank)

第4个client.js是完整的客户端的业务逻辑实现代码，它的内容如下：

[?](http://www.open-open.com/lib/view/open1402479198587.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149 | (function () {      var d = document,      w = window,      p = parseInt,      dd = d.documentElement,      db = d.body,      dc = d.compatMode == 'CSS1Compat',      dx = dc ? dd: db,      ec = encodeURIComponent;          w.CHAT = {          msgObj:d.getElementById("message"),          screenheight:w.innerHeight ? w.innerHeight : dx.clientHeight,          username:null,          userid:null,          socket:null,          //让浏览器滚动条保持在最低部          scrollToBottom:function(){              w.scrollTo(0, this.msgObj.clientHeight);          },          //退出，本例只是一个简单的刷新          logout:function(){              //this.socket.disconnect();              location.reload();          },          //提交聊天消息内容          submit:function(){              var content = d.getElementById("content").value;              if(content != ''){                  var obj = {                      userid: this.userid,                      username: this.username,                      content: content                  };                  this.socket.emit('message', obj);                  d.getElementById("content").value = '';              }              return false;          },          genUid:function(){              return new Date().getTime()+""+Math.floor(Math.random()\*899+100);          },          //更新系统消息，本例中在用户加入、退出的时候调用          updateSysMsg:function(o, action){              //当前在线用户列表              var onlineUsers = o.onlineUsers;              //当前在线人数              var onlineCount = o.onlineCount;              //新加入用户的信息              var user = o.user;                //更新在线人数              var userhtml = '';              var separator = '';              for(key in onlineUsers) {                  if(onlineUsers.hasOwnProperty(key)){                      userhtml += separator+onlineUsers[key];                      separator = '、';                  }              }              d.getElementById("onlinecount").innerHTML = '当前共有 '+onlineCount+' 人在线，在线列表：'+userhtml;                //添加系统消息              var html = '';              html += '<div class="msg-system">';              html += user.username;              html += (action == 'login') ? ' 加入了聊天室' : ' 退出了聊天室';              html += '</div>';              var section = d.createElement('section');              section.className = 'system J-mjrlinkWrap J-cutMsg';              section.innerHTML = html;              this.msgObj.appendChild(section);              this.scrollToBottom();          },          //第一个界面用户提交用户名          usernameSubmit:function(){              var username = d.getElementById("username").value;              if(username != ""){                  d.getElementById("username").value = '';                  d.getElementById("loginbox").style.display = 'none';                  d.getElementById("chatbox").style.display = 'block';                  this.init(username);              }              return false;          },          init:function(username){              /\*              客户端根据时间和随机数生成uid,这样使得聊天室用户名称可以重复。              实际项目中，如果是需要用户登录，那么直接采用用户的uid来做标识就可以              \*/              this.userid = this.genUid();              this.username = username;                d.getElementById("showusername").innerHTML = this.username;              this.msgObj.style.minHeight = (this.screenheight - db.clientHeight + this.msgObj.clientHeight) + "px";              this.scrollToBottom();                //连接websocket后端服务器              this.socket = io.connect('ws://realtime.plhwin.com:3000');                //告诉服务器端有用户登录              this.socket.emit('login', {userid:this.userid, username:this.username});                //监听新用户登录              this.socket.on('login', function(o){                  CHAT.updateSysMsg(o, 'login');              });                //监听用户退出              this.socket.on('logout', function(o){                  CHAT.updateSysMsg(o, 'logout');              });                //监听消息发送              this.socket.on('message', function(obj){                  var isme = (obj.userid == CHAT.userid) ? true : false;                  var contentDiv = '<div>'+obj.content+'</div>';                  var usernameDiv = '<span>'+obj.username+'</span>';                    var section = d.createElement('section');                  if(isme){                      section.className = 'user';                      section.innerHTML = contentDiv + usernameDiv;                  } else {                      section.className = 'service';                      section.innerHTML = usernameDiv + contentDiv;                  }                  CHAT.msgObj.appendChild(section);                  CHAT.scrollToBottom();              });            }      };      //通过“回车”提交用户名      d.getElementById("username").onkeydown = function(e) {          e = e || event;          if (e.keyCode === 13) {              CHAT.usernameSubmit();          }      };      //通过“回车”提交信息      d.getElementById("content").onkeydown = function(e) {          e = e || event;          if (e.keyCode === 13) {              CHAT.submit();          }      };  })(); |

至此所有的编码开发工作全部完成了，在浏览器中打开[http://demo.plhwin.com/chat/](http://demo.plhwin.com/chat/" \t "http://www.open-open.com/lib/view/_blank)就可以看到效果了。

上面所有的客户端和服务端的代码可以从Github上获得，地址：[https://github.com/plhwin/nodejs-socketio-chat](https://github.com/plhwin/nodejs-socketio-chat" \t "http://www.open-open.com/lib/view/_blank)

git clone https://github.com/plhwin/nodejs-socketio-chat.git

下载本地后有两个文件夹 client 和 server，client文件夹是客户端源码，可以放在Nginx/Apache的WebServer中，也可以放在Node.js的WebServer中。后面的server文件夹里的代码是websocket服务端代码，放在Node.js环境中，使用npm安装完 express 和 socket.io 后，node index.js 启动后端服务就可以了。

本例只是一个简单的Demo，留下2个有关项目扩展的思考：

1、假设是一个在线客服系统，里面有许多的公司使用你的服务，每个公司自己的用户可以通过一个专属URL地址进入该公司的聊天室，聊天是一对一的，每个公司可以新建多个客服人员，每个客服人员可以同时和客户端的多个用户聊天。

2、又假设是一个在线WebIM系统，实现类似微信，qq的功能，客户端可以看到好友在线状态，在线列表，添加好友，删除好友，新建群组等，消息的发送除了支持基本的文字外，还能支持表情、图片和文件。