**Documentation to use this node.js server for create an istant messaging chat service.**

To use this plug in you have to know websocket API.

First of all download *server.js* file and run it on a node js server (I tried it on nodejitsu.com, for example).   
After the server has been launched, start to interact with it, on the client side.

The first method of this plug-in that you will need to use is jQuery.connect = function(address){.. }

This method takes in input the address of the nodejs server e enstablish a connection. To use our server you need to invoke connect function in this way:  
  
*$.connect('ws://uniswerchat.jit.su'); // You have to put the adress of your server.*

Once you have enstablished the connection, you probably want to communicate to the server that a new client is online. So you can call   
  
*jQuery.addUser = function(user\_id, contact\_list){...}*  
  
This function takes in input an user identificator (that will be the only way to refer to it on the server-side.) and the list of the users identificators whereby we want to connect himself.

At this point you may want to send a notification, send a message in a one-one chat or send the typing event.

To send a notification you can use the function

*jQuery.sendNotification = function(html, receivers\_list ){..}*

You simply have to pass to this function the html code as a string. In this way on the others client side that will receive the notification, you can simply append() the notification where you want.

You need also to specify the list of users identificators that need to receive the notification.

To send a textual message in a one-one chat you have to call

*jQuery.sendMess = function(author, receiver, text){..}*

You need to pass:

*author =* user identificator,

*receiver =* user identificator that will receive the message.

*text =* content of the message.

Finally if you want send the typing (and no\_typing) event you have to call

*jQuery.typing = function(author, receiver, event){...}*

You need to pass:

*author =* user identificator,

*receiver =* user identificator that will receive the message.

*event =* it can assume two value 'typing' and 'no\_typing'.

At the moment multiple conversations are not supported.

To handling the incomings message on the client-side you first of all need to use the websocket API listener *connection.onmessage = function(message){... }.*

To dinstinguish among the different event message that can arrive you can use this simple code:

*try {*

*var json = JSON.parse(message.data);*

*} catch (e) {*

*console.log('This doesn\'t look like a valid JSON: ', message.data);*

*return;*

*}*

*switch(json.type){*

*case 'notification':*

*$('#'+element).append(json.html);*

*alert("NOME :" + json.html);*

*break;*

*};*

Messages send by the server are in the JSON form. To read correctly the content of the message you need to parse the message.   
  
Then you can handle different messages type using switch condition on the field *type of the message.*

Possible events are:

- notification

- update\_user

- new\_name

- utenti\_connessi

- typing

- no\_typing

- testo

- off\_line ??

- on\_line ??

- user\_off

***notification***

this event means the arrive of a notification.

the message contain this structure:

*var config = {*

*type: 'notification',*

*html: html,*

*receivers: receivers\_list*

*};*

***update\_user***

This event communicate that a new user is online.

the message contain this structure:

*var struct = {*

*type:'update\_user',*

*utenti:*

*};*

In the field utenti there is the user identificator that has just connected

***new\_name***

This event is used to handle the opening of multiple browser tab by the same user.

So when a user enstablish a connection and notify his id to the server, this one send to the new user a new identificator whereby the programmer have to replace the initial identificator, so use the last identificator for the next operation.

the message contain this structure:

*var new\_name= {*

*type: 'new\_name',*

*nome:*

*};*

***utenti\_connessi***

This event notify, after the user enstablish a connection, the others users that are already online.

The structure is:

*var info\_utenti\_on = {*

*type:'utenti\_connessi',*

*utenti:*

*};*

The fild utenti contains a list of the online users

***typing***

Event that notify the typing.

The structure is:   
  
  
*var struct = {*

*type: 'typing',*

*autore: ,*

*destinatario:*

*};*

The field autore contains the identification of the user that is writing.

The field destinatario conatins the receivers of this event.

***no\_typing***

Event that notify the stop to typing.

The structure is:   
  
  
*var struct = {*

*type: 'no\_typing',*

*autore:,*

*destinatario:*

*};*

The field autore contains the identification of the user that is writing.

The field destinatario contains the receivers of this event.

***testo***

Event that notify the arrive of a textual message.

*var struct = {*

*type:'testo',*

*autore: ,*

*destinatario: ,*

*testo:*

*};*

The field autore contains the identification of the user that is writing.

The field destinatario contains the receivers of this event.

The field testo contains the text of the message.

***user\_off***

This event notify that a user is no longer online.

*var utente\_off = {*

*type: 'user\_off',*

*user:*

*};*

user field contains the identificator of the user that is passed offline.