Online chat application form:

Lets do this by using node js, socket.io and jquery.

Part-1:The basics just sending data to server and getting back again..phew that’s interesting

Step:1

Create a package.json file

Ex:

{

"name":"chat",

"version":"0.0.1",

"private":"true",

"dependencies":{

"socket.io":"1.7.2", ------------type in cmd as npm info socket.io version

"express":"4.14.0" --------------------------------- npm info express version

}

}

Lets make the folder as chat in that type npm install express it downloads the node modules folder to it.

Lets create app.js,index.html files for use

Now in app.js we have

var express=require("express"),

app=express(),

server=require('http').createServer(app), -------it creates server for our application

io=require('socket.io').listen(server), ----------------it creates socket.io things for us

server.listen(8000); ------------installing server on this address

console.log("started?");

app.get('/',function(req,res){ --------------here we are creating a route for our page

res.sendfile(\_\_dirname+'/index.html');

});

Now on our index file lets create a div where all messages will go and a form where we can type message.

After that now coming to our jquery side:

2 source codes for jquery and socket.io to work on

<script src="http://code.jquery.com/jquery-latest.min.js"></script>

<script src="/socket.io/socket.io.js"></script>

jQuery(function($){ -----------this is basically a on ready function

var socket=io.connect(); -------------setting up socket.io

var $messageForm=$('#send-message');------------this is form id

var $messageBox=$('#message');----------------------this is input type message

var $chat=$('#chat'); --------------------------this is our div where our messages are going

}

$messageForm.submit(function(e){ -------------------this happens when we submit our form

e.preventDefault(); -------------------this will prevent loading of the page when submits

});

Now we need to send our value to the server right..?? yup for that we need this

socket.emit('send message',$messageBox.val()

here ‘send message’ is an event happening when we click on submit.the value is $messageBox.val().

Now we need to receive the value from our index page lets see how to do that

io.sockets.on('connection',function(socket){ ------------this for all our code will be stored here

}

This code will be interesting here--

socket.on('send message',function(data){

io.sockets.emit('new message',data) ----------------here we are emitting the data to the all users

}

Wow the server receives the data,it again send back the data to the index page,to recevive that we need to do---

socket.on('new message',function(data){

$chat.append(data) --------------here we are appending the data to the div column

});

By the end of this in our page we can see we can send message and that message is displaying on div.

Part:2 Adding usernames:

Now we will check whether the username is valid or not.ie new users cant take the name of the users who are already in..!!

Consider

$nickForm.submit(function(e){ -------------this happes when u click on the username submit btn

e.preventDefault();

socket.emit('new user',$nickBox.val(),function(data){------------sending data to server

}-------------------function(data) gives us the callback fn to wat we are sending

$nickBox.val('');

})

Now on the server side we need to check that username is going to be unique

socket.on('new user',function(data,calback){

if(nicknames.indexOf(data)!=-1){ -------------here we are checking that nicknames are unique

callback(false); ------nicknames[] array saving names

} ----------negative values indicate that somewhere username is exists

});

Else{

Callback(true);-----if its true then we can send data bak to index page

socket.nickname=data;-----if the username is unique we are storing that in some variable

nicknames.push(socket.nickname);----and later we adding that to our array

io.sockets.emit(‘username’,nicknames) ------now here we are sending data bak again to indexpage

}

Now coming bak to index page

socket.emit('new user',$nickBox.val(),function(data){ ----data is the value from server

if(data){----if data is true then if condition true

$('#nickWrap').hide();

$('#contentWrap').show();

}

else{

$nickError.html("username already taken! Try again");

}

})

So finally we checked here whether the user is unique or not..good step right? …now we nee dtop display the names of the users who are logged in on the right side..

Now we will display the names of the users on the right side:

socket.on('usernames',function(data){

var signedusers="";

for(var i=0;i<data.length;i++){ ---------here the data is the array in server side which is storing the names

signedusers +=data[i]+'<br>'

}

$users.html(signedusers); ------now we are adding newsigned users on the rightside

});

Now we can see the users on the right side.but here the problem is eventhough the user exits still we can see his/her name on the list.so lets remove it

So for the users who are disconnected here we have:

socket.on('disconnect',function(data){

if (!socket.nickname) { ----it indicates the user must use username to login

return

}

Nicknames.splice(nicknames.indexOf(socket.nickname),1)----here we are removing the users who logout

updateNicknames() ----------this is a functions that automatically updates the nicknames list

})

Function updateNicknames() {

io.sockets.emit(‘username’,nicknames) ----here we are sending the updated list to the index page

}

socket.on('new user',function(data,callback){

if(data in users){

callback(false);

}

else{

callback(true);

socket.nickname=data;

updateNicknames();----------here it is the extra added function to previous one

}

});

Yes for of now it will works superbly---the logout users are just disappearing from our box

Now the most important step—displaying the name of the user who sends the message----!!!

This step is pretty easy…when we sending the messages from the server side

io.sockets.emit('new message',{msg:msg,nick:socket.nickname}---here msg is the message that we type and nick is the name of the user

now on index page just update it as

$chat.append('<span class="msg"><b>'+data.nick+':'+data.msg+'</span><br>');

Now its working superblyyy…now we have add one more point:

How about a private chat..???

Step:3-private chat:

This is bit tricky…hold on

Here the nicknames array which is on server side changed to object with name users

Now major steps coming in..just a fe touch ups that it

1.in new user function change

if(nicknames.indexOf(data)!=-1){ ----------- to

if(name in users)

}

2.

Nicknames.splice(nicknames.indexOf(socket.nickname),1)---- to

delete users[socket.nickname];

here the major step is saving every nickname of the users in socket.This socket is going to be acts as the reference to ours

its time then…

if they want to whisper they need to use ‘/w ‘ before so lets check whether thay are using it or not

so to remove spaces lets trim our msg in send message section:

var msg=data.trim();

if(msg.substring(0,3)==='/w '){ ----if it happens then they are whispering

now we will see who are whispering:

var ind=msg.indexOf(' ');

if(ind !== -1){ ----this is for seeing there is any msg

else{ callback('Error! Please enter a message to whispher');--------if not then display this

so when the msg is valid we will check whether the username is valid or not—good step

var name=msg.substring(0,ind);---here the name starts before message which starts from 3

var msg=msg.substring(ind + 1);-----it indicares the message

so after entering ur name we have to check whether the person is in users or not

if(name in users){

users[name].emit('whisper',{msg:msg,nick:socket.nickname});

}else{

callback('Error! Please enter valid user');

}

Now on client side add socket.whispher to see

socket.on('whisper',function(data){

$chat.append('<span class="whisper"><b>'+data.nick+':'+data.msg +'</span><br>');

})

Yes that it..private chat is done.

Now we wills e how to add all these msgs in database

Will use Mongodb for that…

Step:4 adding data to database: