CHAT APPLICATION

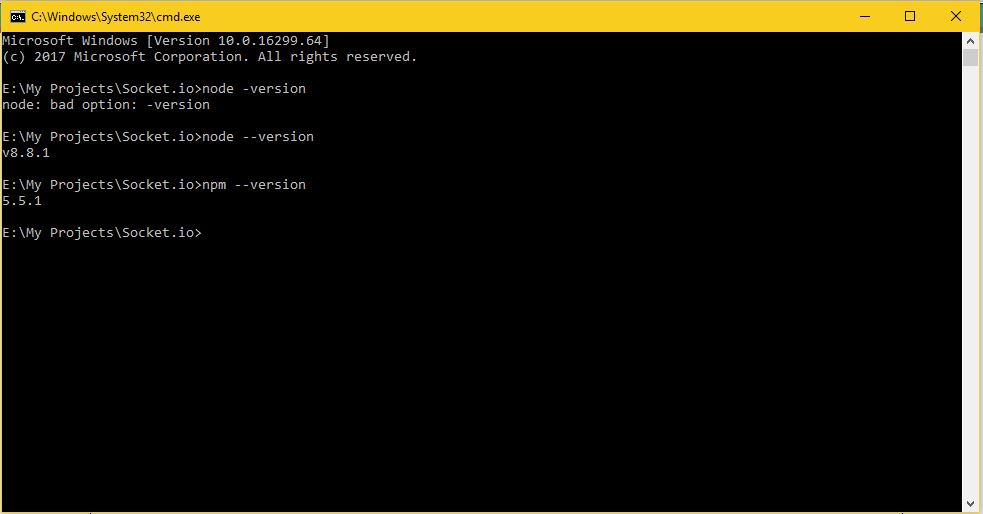
Author: Sambit Kumar Padhy

To get started with developing using the **Socket.IO**, you need to have **Node** and **npm (node package manager)** installed. If you do not have these, head over to **Node setup** to install node on your local system.

Confirm that node and npm are installed by running the following commands in your terminal.

node --version

npm --version

You should get an output similar to: 

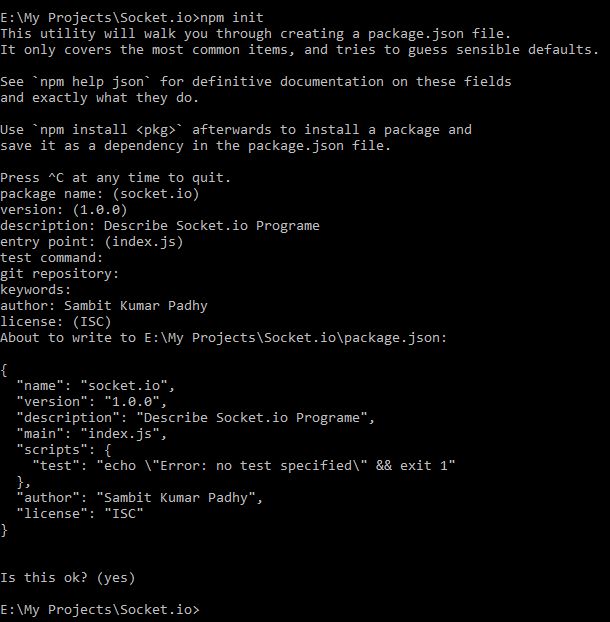
Open your terminal and enter the following in your terminal to create a new folder and

enter the following commands:

$ mkdir Socket.io

$ cd Socket.io

$ npm init

It will ask you some questions; answer them in the following way: 

This will create a ‘**package.json node.js’** configuration file. Now we need to install **Express** and **Socket.IO**. To install these and save them to **package.json** file, enter the following command in your terminal, into the project directory:

npm install --save express socket.io

One final thing is that we should keep restarting the server. When we make changes, we

will need a tool called **nodemon**. To install nodemon, open your terminal and enter the

following command:

npm install -g nodemon

Whenever you need to start the server, instead of using the **node app.js** use, **nodemon app.js**. This will ensure that you do not need to restart the server whenever you changea file. It speeds up the development process.Now, we have our development environment set up. Let us now get started withdeveloping real-time applications with Socket.IO.

Now that we are well acquainted with Socket.IO, let us write a chat application, which we can use to chat on different chat rooms. We will allow users to choose a username and allow them to chat using them. So first, let us set up our HTML file to request for a username.

We will need an **index.html** file to serve, create a new file called index.html and enter the following code in it:

<!DOCTYPE html>

<html>

<head><title>Hello world</title></head>

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

<script>

var socket = io();

</script>

<body>

<input type="text" name="name" value="" placeholder="Enter your name!">

<button type="button" name="button">Let me chat!</button>

</body>

</html>

Now that we have set up our HTML to request for a username, let us create the server **app.js** file to accept connections from the client. We will allow people to send their chosen usernames using the **setUsername** event. If a user exists, we will respond by a **userExists** event, else using a **userSet** event.

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

var http = require('http').Server(app);

var io = require('socket.io')(http);

app.get('/', function(req, res){

res.sendfile('index.html');

});

users = [];

io.on('connection', function(socket){

console.log('A user connected');

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

if(users.indexOf(data) > -1){

users.push(data);

socket.emit('userSet', {username: data});

}

else{

socket.emit('userExists', data + ' username is taken! Try some other username.');

}

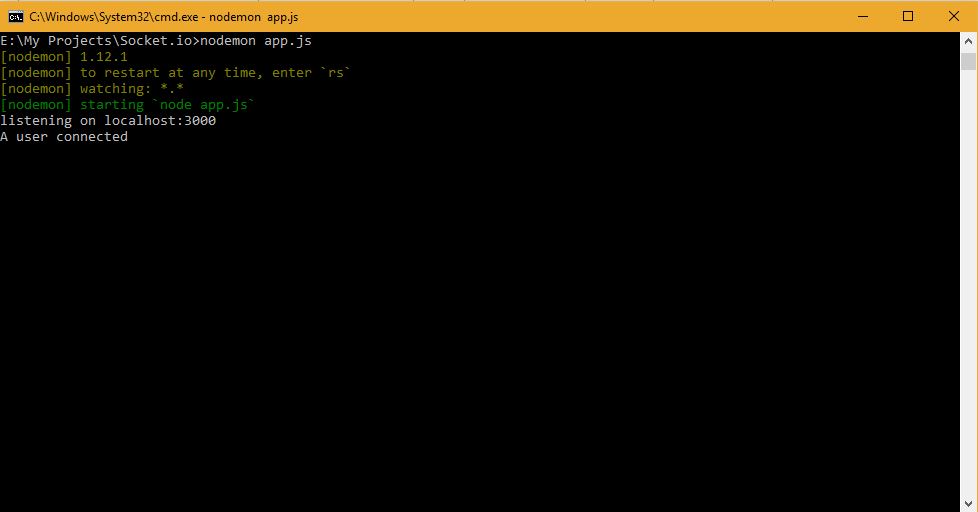
})

});

http.listen(3000, function(){

console.log('listening on localhost:3000');

});



Type in web browser localhost:3000 and press enter…..It .

C:\Users\Sambit\Desktop\socket.io\Capture6.JPG

We need to send the username to the server when people click on the button. If user

exists, we show an error message; else, we show a messaging screen:

<!DOCTYPE html>

<html>

<head><title>Hello world</title></head>

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

<script>

var socket = io();

function setUsername(){

socket.emit('setUsername', document.getElementById('name').value);

};

var user;

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

document.getElementById('error-container').innerHTML = data;

});

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

user = data.username;

document.body.innerHTML = '<input type="text" id="message">\

<button type="button" name="button" onclick="sendMessage()">Send</button>\

<div id="message-container"></div>';

});

function sendMessage(){

var msg = document.getElementById('message').value;

if(msg){

socket.emit('msg', {message: msg, user: user});

}

}

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

if(user){

document.getElementById('message-container').innerHTML +=

'<div><b>' + data.user + '</b>: ' + data.message + '</div>'

}

})

</script>

<body>

<div id="error-container"></div>

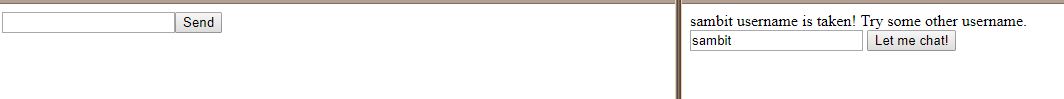
<input id="name" type="text" name="name" value="" placeholder="Enter your name!">

<button type="button" name="button" onclick="setUsername()">Let me chat!</button>

</body>

</html>

Now if you connect two clients with same username, it will give you an error as shown in

the screenshot below: 

Once you have provided an acceptable username, you will be taken to a screen with a

message box and a button to send messages. Now, we have to handle and direct the

messages to the connected client. For that, modify your app.js file to include the following

changes:

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

var http = require('http').Server(app);

var io = require('socket.io')(http);

app.get('/', function(req, res){

res.sendfile('index.html');

});

users = [];

io.on('connection', function(socket){

console.log('A user connected');

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

console.log(data);

if(users.indexOf(data) > -1){

socket.emit('userExists', data + ' username is taken! Try some other username.');

}

else{

users.push(data);

socket.emit('userSet', {username: data});

}

});

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

//Send message to everyone

io.sockets.emit('newmsg', data);

})

});

http.listen(3000, function(){

console.log('listening on localhost:3000');

});

Now connect any number of clients to your server, provide them a username and start

chatting! In the following example, we have connected two clients with names sambit and

Akhil and sent some messages from both the clients:

