Chat application using Node.js and Socket.io

Purpose and Evolution of the idea

The main purpose is to create a chat application which provides a common platform for users to interact and discuss. Not so long ago, if we wanted to build a simple JavaScript chat we had to work with Ajax, Comet or Flash component in the browser.

Ajax can be a little bad for performance, (in the case of chatting) because it's based on a request and response. We also needed to hold an interval in the client which asked the server for new messages and this process was repeated again and again.

Comet is pretty much the same. While using Ajax, we asked the server every time for messages, with Comet we needed to open one connection and wait. This connection remained open in the server for a while until the server received messages for us, then the server responded with the messages and closed the opened connection. The client, when receiving these messages opened a new connection and waited again. One advantage of Comet is the reduction of the request amounts, but still, we needed to open and close many of them for each session (in case the user is active).

And lastly, a Flash component - in this case we placed a small flash component in our page to exchange messages with the JavaScript client, the flash could open a socket to the server and this gave us push communication.

Node.js and Socket.IO

With node.js and socket.io we can enjoy better performance, a bidirectional push communication between a server and a JavaScript client and even the support for all the browsers. Socket.io can work with several transports in order to support even old browsers. NodeJS provides the ability to write back-end code in JavaScript which is the perfect technology for building real time applications.

Technology, Tools used

1. MEAN Stack (Express, Angular, Node)
2. Sockets to enable one-on-one communication in real time- We will set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server side. Any client that has a socket associated with the same port can communicate with the server socket.

Future Prospect of the project:

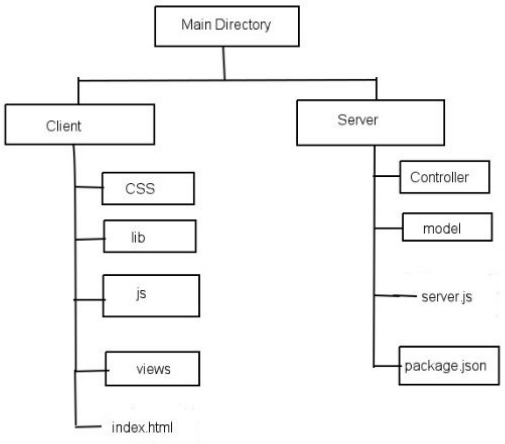
We are also planning to deploy the use of HTML5 APIs for real-time communications such as canvas/video, sockets, getUserMedia, and WebRTC. Along with it we wish to dive into Node.js and learn all about how to use it from the ground up in the command line to communicating with HTML5 in real-time through asynchronous code in Node.js. We also wish to mix up our chat application with Google Maps and the Geolocation API to create a cool geographic app.

Dependencies-

- socket.io — is a javascript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers. The client-side library runs in the browser and the server-side library for node.js. Both components have similar API & like node.js, it is also event-driven. Socket.IO provides the ability to implement real-time analytics, binary streaming, instant messaging, and document collaboration.

- express — is a Node.js web application framework. It provides the set of features to develop the web and mobile applications. One can respond to HTTP request using different middlewares and also render HTML pages.

Directory structure and the path to be followed ahead-



Directory structure of the project

• In NodeJS command prompt/terminal we will navigate to our application directory i.e. chat-application.

• Create a package.json file which is the manifest file that describes our project.
• Install Express, HTTP, socket.io and path npm modules in our working directory i.e. chat-application.
• Create an HTML file index.html which is used to serve the chat window in the browser.
• Create a CSS file i.e. style.css to design the chat window.
• Next step is to create an index.js file that will setup our application on the server.
• Then we will create file for client side script i.e. chat.js.
• Go to your web browser and then open link in two different windows to test our chat application

Index.js code

```
var app = require('http').createServer(handler),
    io = require('socket.io').listen(app),
    fs = require('fs')

 app.listen(80);

function handler(req, res) {
    fs.readFile(__dirname + '/index.html',
        function (err, data) {
            if (err) {
                res.writeHead(500);
                return res.end('Error loading index.html');
            }

            res.writeHead(200);
            res.end(data);
        });
}

io.sockets.on('connection', function (socket) {
    socket.emit('news', {
        hello: 'world'
    });
    socket.on('my other event', function (data) {
        console.log(data);
    });
});
```