SIT 725 Prac 7
Sockets

Contents

- Sockets
- Adding socket to NodeJs Application
- Questions

Sockets

While it is a wrapper around WebSockets For Node.js, Socket.io is a Library for Connecting a Client(s) to a Server utilising the Client/Server Architecture. It is incredibly easy and simple to use, especially when dealing with chat messages or real-time data.

A socket is a single connection between a client and a server that allows both the client and the server to send and receive data at the same time. Because Library is an event-driven system, it emits and listens for certain events to be triggered.

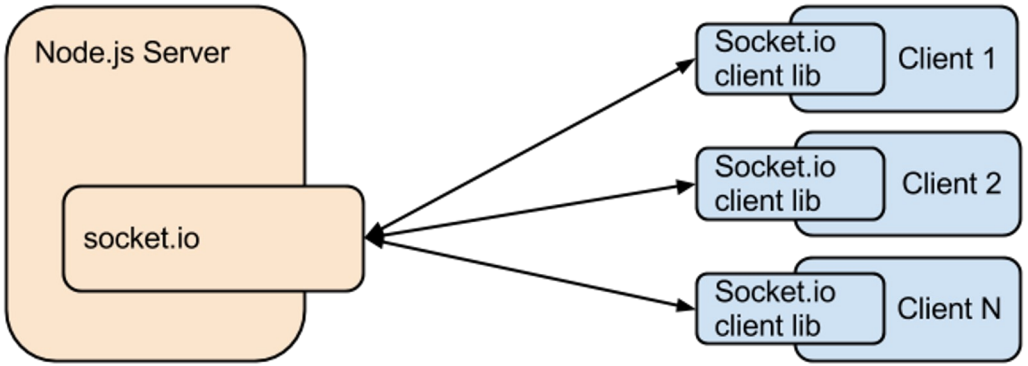
Sockets Cont ...

So let's try to understand how our application design would look like when we add socket.io to our application.

From the image what we can understand is that the nodejs server hosts one instance of socket.io serv side package and every client host its own instance of socket.io client. So we can get to know that the server does not need multiple hosts to handle multiple clients.

This means that it is a one to many relation where one server side socket.io can handle multiple client side socket connections.

While on the other hand the client can at one point only listen to one server side socket connection.



Adding sockets to NodeJs Application

Now For creating the socket.io application, we need to install the socket.io npm to our application and embed it into our application so that we have a server side socket ready to use.

Installing the socket.io library

```
$ npm install socket.io --save
```

Now that we have the socket.io npm installed. Lets see what we need to add in our application inorder to connect it to a client side application.

Adding sockets to NodeJs Application Cont …

In order to add socket.io library to our server.js file we need to add a little bit of code to our application.

```
const http = require('http').createServer(app); // Create HTTP server from app
const io = require('socket.io')(http); // Pass http server to socket.io

….
…
io.on('connection', (socket) => {
  console.log('a user connected');
  socket.on('disconnect', () => {
    console.log('user disconnected');
  });
  setInterval(()=>{
    socket.emit('number', parseInt(Math.random()*10));
  }, 1000);

});
…
…
```

Make sure the http object is created using your Express app and passed to socket.io. If you use let io = require('socket.io')(http); without defining http, you will get an error.

Adding sockets to NodeJs Application Cont ...

Let's try to understand what is happening in that code.

So first we require the library in our application, and then we create a listener that whenever a user connects it prints in the console that the user has connected and also it has a set interval function that sends a random number on an event name called number to all the users that are connected to our socket server.

A point to notice is that the socket library always works on the basis of events the data is always sent on event name and read using the help of these event names on the client side. This gives the ability to help us listen to different event names and also perform different actions on the client side using those event names.

Adding sockets to NodeJs Application Cont ...

*Always load socket.io.js **before** your scripts.js or any custom JS files that use the io object.*

Let's try to add the socket library to our client side.

If you see the index.html file of our client application you can see that first we need to add the socket library to our index.html file we can do that by adding this line of code.

```
<!-- Load socket.io BEFORE scripts.js -->
<script src="/socket.io/socket.io.js"></script>
<script src="scripts.js"></script>
```

We add the script which is reading the socket file from our socket.io folder installed using the npm and we will use this socket library in our scripts.js file. Now let's add the socket event to read the data from our socket library from the server.

Adding sockets to NodeJs Application Cont ...

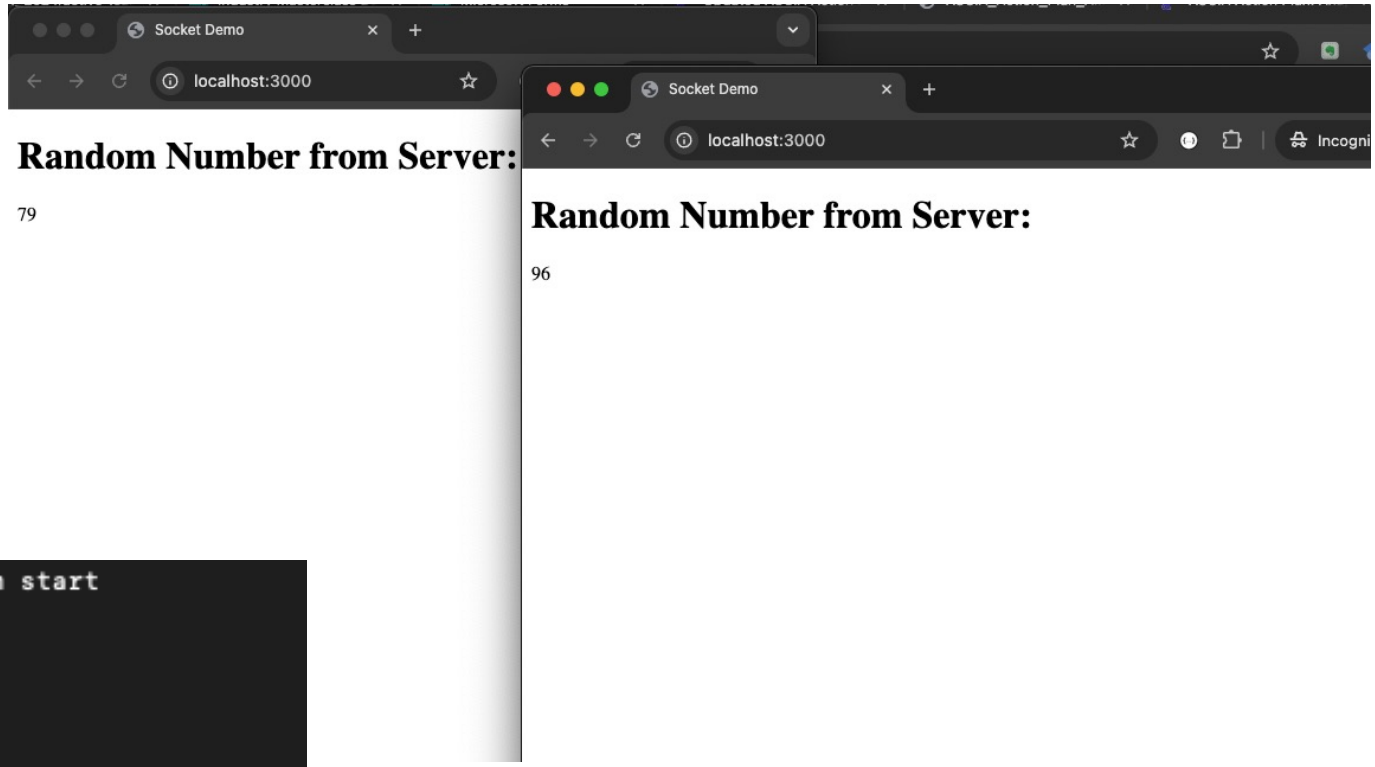
Next we update our scripts.js file to listen to events

```
// Listen to 'number' event
socket.on('number', (msg) => {
console.log('Random number:', msg);
document.getElementById('number').innerText = msg;
});
```

Here we create a socket variable using the io library and then we tell our application that the moment it connects to our socket start listening to the ‘number’ event and whatever msg you get from the server display it.

Adding sockets to NodeJs Application Cont …

Now that we have added the code both side lets see if our application behave as we expect. So we start our application.




```
[niroshinie.fernando@DCCP20R2XG week7_1 % npm start

> week7_1@1.0.0 start
> node server.js

Server running at http://localhost:3000
A user connected
A user connected
A user connected
```

Source: https://github.com/niroshini/sit725/tree/main/week7_1

Adding sockets to NodeJs Application Cont ...

So we can see that our application prints whenever we get a user connected to our application and also on the client side we are receiving an random number from the socket library that is being broadcasted by our server. This is the exact result we were hoping to see.

So now you have connected a socket system to your application.

You can also use more things from the socket library for doing that please read the document provided by the socket.io.

Example Use Cases in Web Apps

A few ideas…

- E-commerce App:
  - Live stock updates when another user purchases an item
  - Real-time order status tracking ("Packing", "Shipped", etc.)
  - Live chat support between customers and vendors
- IoT Dashboard:
  - Sensor data updates on the dashboard in real-time

Thanks