Alunos: José Raimundo Fernandes
José Victor Dantas
Professor: Gustavo Wagner

Objetivo
Nesta atividade você construirá uma aplicação de chat em javascript utilizando bibliotecas como Express e Socket.io

Introdução
Os sockets têm sido tradicionalmente a solução em torno da qual a maioria dos sistemas de bate-papo em tempo real são arquitetados, fornecendo um canal de comunicação bidirecional entre um cliente e um servidor.

Isso significa que tanto o cliente quanto o servidor podem enviar mensagens aos clientes a partir de um evento. O servidor pode, também, enviar para todos os outros clientes conectados.

Passos
Antes de começar, tenha certeza que o NodeJs está instalado em sua máquina

1. Crie uma pasta chamada “chat-example”
2. Abra o terminal nesta pasta e digite “npm init”, sem aspas
3. Agora digite “npm install express@4.15.2” sem aspas
4. Crie um arquivo chamado “index.js” e cole o conteúdo abaixo

```
var app = require('express')();
var http = require('http').createServer(app);

app.get('/', (req, res) => {
 res.send('<h1>Hello world</h1>');
});

http.listen(3000, () => {
 console.log('listening on *:3000');
});
```

   a. Express inicializa a variável “app” para então ser fornecida na criação do servidor http
   b. Definimos a rota “/” que servirá como página inicial
   c. O servidor http escutará na porta 3000
5. Inicie o servidor com o comando “node index.js” e digite “http//:localhost:3000” no navegador e veja Hello World escrito
6. Refatore a rota no index.js para usar o método sendFile()

```
app.get('/', (req, res) => {
 res.sendFile(__dirname + '/index.html');
});
```

7. Crie um arquivo index.html e coloque:

```
<!doctype html>
<html>
 <head>
   <title>Socket.IO chat</title>
   <style>
     * { margin: 0; padding: 0; box-sizing: border-box; }
     body { font: 13px Helvetica, Arial; }
     form { background: #000; padding: 3px; position: fixed; bottom: 0;
width: 100%; }
     form input { border: 0; padding: 10px; width: 90%; margin-right:
0.5%; }
     form button { width: 9%; background: rgb(130, 224, 255); border:
none; padding: 10px; }
     #messages { list-style-type: none; margin: 0; padding: 0; }
     #messages li { padding: 5px 10px; }
     #messages li:nth-child(odd) { background: #eee; }
   </style>
 </head>
 <body>
   <ul id="messages"></ul>
   <form action="">
     <input id="m" autocomplete="off" /><button>Send</button>
   </form>
 </body>
</html>
```

8. Pare o servidor e inicie novamente com “node index.js” e recarregue a página no navegador

Socket.IO é composto em duas partes:

- Um servidor que se integra com socket.io do servidor HTTP Node.JS
- Uma biblioteca cliente que carrega no lado do navegador socket.io-client

9. Instale o socket.io com o comando “npm install socket.io” sem aspas
10. Agora edite o index.js

```
var app = require('express')();
var http = require('http').createServer(app);
var io = require('socket.io')(http);

app.get('/', (req, res) => {
 res.sendFile(__dirname + '/index.html');
});

io.on('connection', (socket) => {
 console.log('a user connected');
});

http.listen(3000, () => {
 console.log('listening on *:3000');
});
```

    a. Foi inicializada a instancia do socket.io passando como parâmetro o objeto http
    b. Então ele escuta pelo evento “connection” que espera a chegada de novos sockets para imprimir na tela “a user connected”

11. No index.html, logo abaixo da tag </body> adicione:

```
<script src="/socket.io/socket.io.js"></script>
<script>
 var socket = io();
</script>
```

    a. Isso é tudo o que é necessário para carregar o socket.io-client.

12. Adicione agora o evento de “disconnect” aninhado ao evento “connect”:

```
io.on('connection', (socket) => {
 console.log('a user connected');
 socket.on('disconnect', () => {
   console.log('user disconnected');
 });
});
```

13. Reinicie a aplicação e veja o resultado do console.log

A ideia principal do Socket.IO é que você possa enviar e receber todos os eventos que desejar, com os dados que desejar. Qualquer objeto que possa ser codificado como JSON e também como Dados Binários podem ser transportados.

14. No index.html, na parte de script adicione o seguinte trecho de código para emitir um evento chamado “chat message”:

```
<script src="/socket.io/socket.io.js"></script>
<script
src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script>
 $(function () {
   var socket = io();
   $('form').submit(function(e) {
     e.preventDefault(); // prevents page reloading
     socket.emit('chat message', $('#m').val());
     $('#m').val('');
     return false;
   });
 });
</script>
```

15. No index.js, nós escutamos o evento “chat message” e imprimimos no console:

```
io.on('connection', (socket) => {
 socket.on('chat message', (msg) => {
   console.log('message: ' + msg);
 });
});
```

O próximo passo é emitir um evento para todos os usuários

16. Edite o código anterior da seguinte forma:

```
io.on('connection', (socket) => {
 socket.on('chat message', (msg) => {
   io.emit('chat message', msg);
 });
});
```

17. No lado do cliente, capturamos o evento e incluímos o resultado na página:

```
<script>
 $(function () {
   var socket = io();
   $('form').submit(function(e){
     e.preventDefault(); // prevents page reloading
     socket.emit('chat message', $('#m').val());
     $('#m').val('');
     return false;
   });
   socket.on('chat message', function(msg){
     $('#messages').append($('<li>').text(msg));
   });
 });
</script>
```

18. Agora abra duas janelas diferente e converse consigo mesmo

Desafios:

- Transmita uma mensagem para usuários conectados quando alguém se conectar ou desconectar.
- Adicionar suporte para apelidos
- Adicione a funcionalidade “{usuário} está digitando”.
- Mostre quem está online.