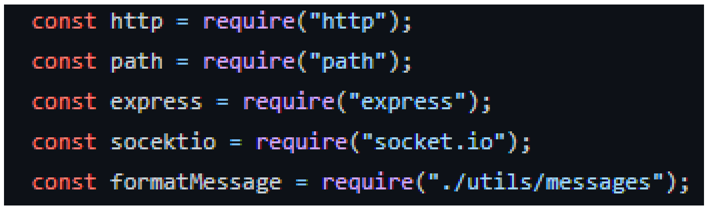
Code Challenge

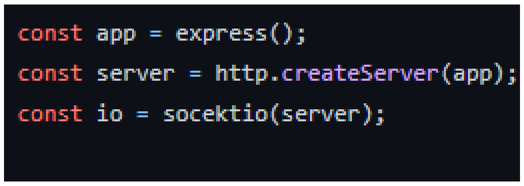
I developed a real-time chat-application using socket.io, Node.js and Express with Vanilla.js on the frontend.

Server.js:

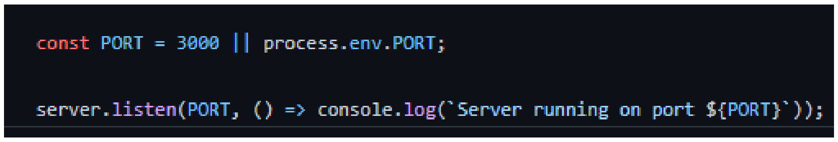
- I started with setting up the server for the application and installed all the npm packages such as express, socket.io etc. Socket.io is a library that enables real-time, bidirectional and event based connection between the browser and server



```
const http = require("http");
const path = require("path");
const express = require("express");
const socektio = require("socket.io");
const formatMessage = require("./utils/messages");
```

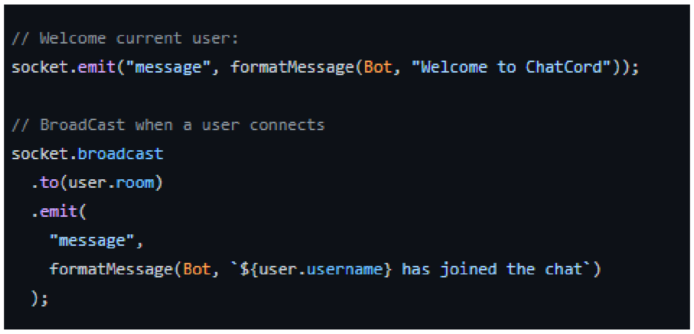


```
const app = express();
const server = http.createServer(app);
const io = socektio(server);
```



```
const PORT = 3000 || process.env.PORT;

server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

- As soon as the client connects to the server, two messages are broadcasted to the chat room.
    - “Welcome to ChatCord”.
    - “Alex has joined the chat”.



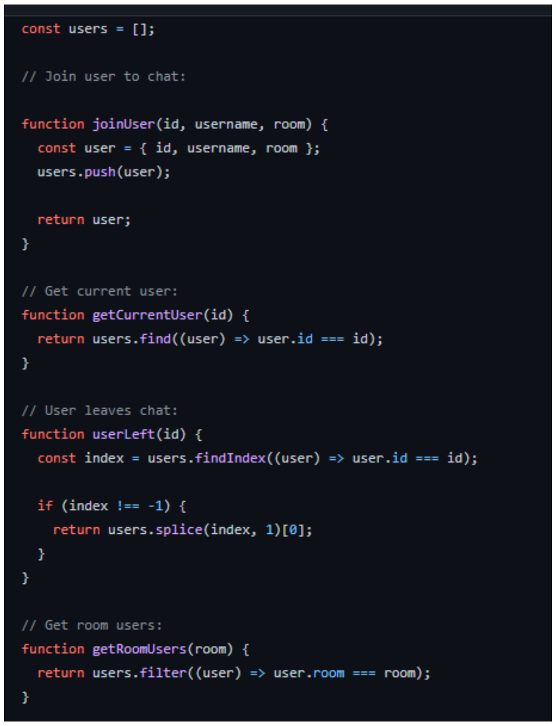
```
// Welcome current user:
socket.emit("message", formatMessage(Bot, "Welcome to ChatCord"));

// BroadCast when a user connects
socket.broadcast
  .to(user.room)
  .emit(
    "message",
    formatMessage(Bot, `${user.username} has joined the chat`)
  );
```

- When a client leaves the chat, a message is broadcasted that a user has left the chat.

User.js:

- I created a separate file for users which handles all the functionalities of a user such as joining the **server**, **leaving the server**, **getting the current user** etc.

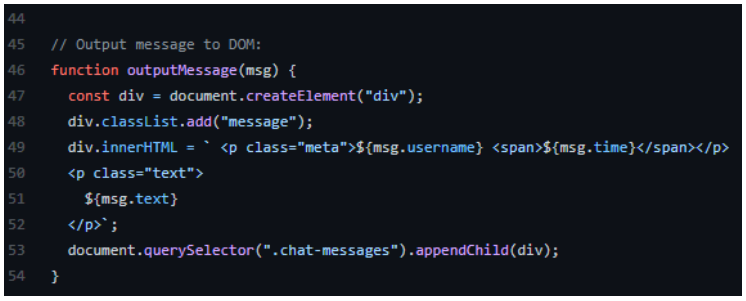
- For the challenge, instead of using databases, I stored the user information in an array called **users.**
- Every time a new user enters a chatroom, an object is created which contains the information : **username, id, room.** This object is then **pushed** to the **users** array.



```
const users = [];

// Join user to chat:

function joinUser(id, username, room) {
  const user = { id, username, room };
  users.push(user);

  return user;
}

// Get current user:
function getCurrentUser(id) {
  return users.find((user) => user.id === id);
}

// User leaves chat:
function userLeft(id) {
  const index = users.findIndex((user) => user.id === id);

  if (index !== -1) {
    return users.splice(index, 1)[0];
  }
}

// Get room users:
function getRoomUsers(room) {
  return users.filter((user) => user.room === room);
}
```

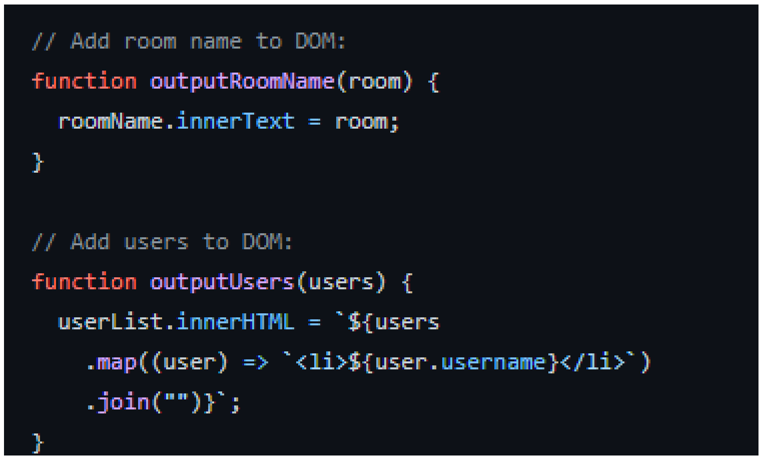
Main.js:

It is the client side javascript file which acts as an interface between the server and the user.
- Outputs the message to the DOM.



```javascript

// Output message to DOM:
function outputMessage(msg) {
  const div = document.createElement("div");
  div.classList.add("message");
  div.innerHTML = ` <p class="meta">${msg.username} <span>${msg.time}</span></p>
  <p class="text">
    ${msg.text}
  </p>`;
  document.querySelector(".chat-messages").appendChild(div);
}
```

- Add room names and usernames to the DOM.

  - 

```javascript
// Add room name to DOM:
function outputRoomName(room) {
  roomName.innerText = room;
}

// Add users to DOM:
function outputUsers(users) {
  userList.innerHTML = `${users
    .map((user) => `<li>${user.username}</li>`)
    .join("")}`;
}
```