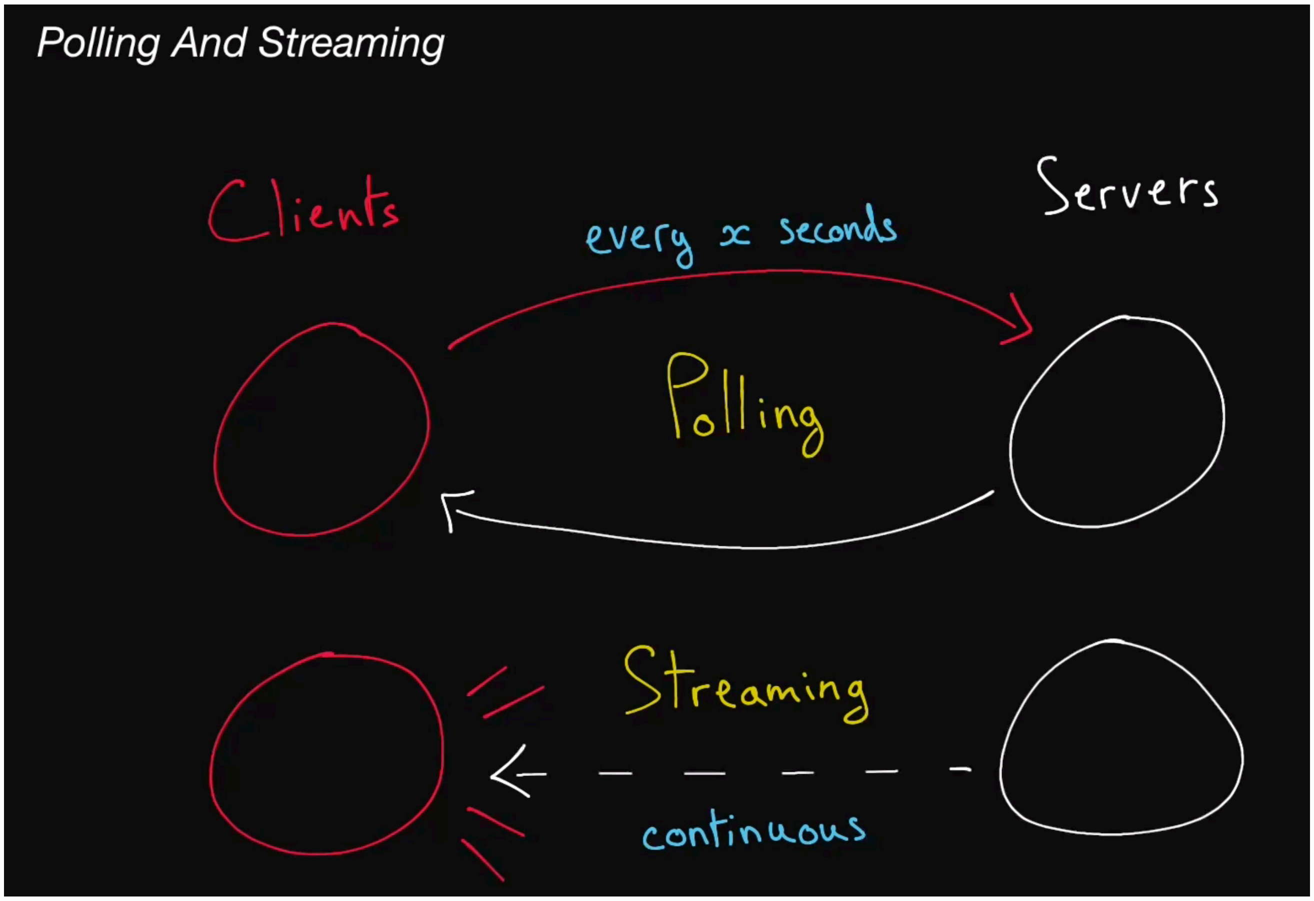
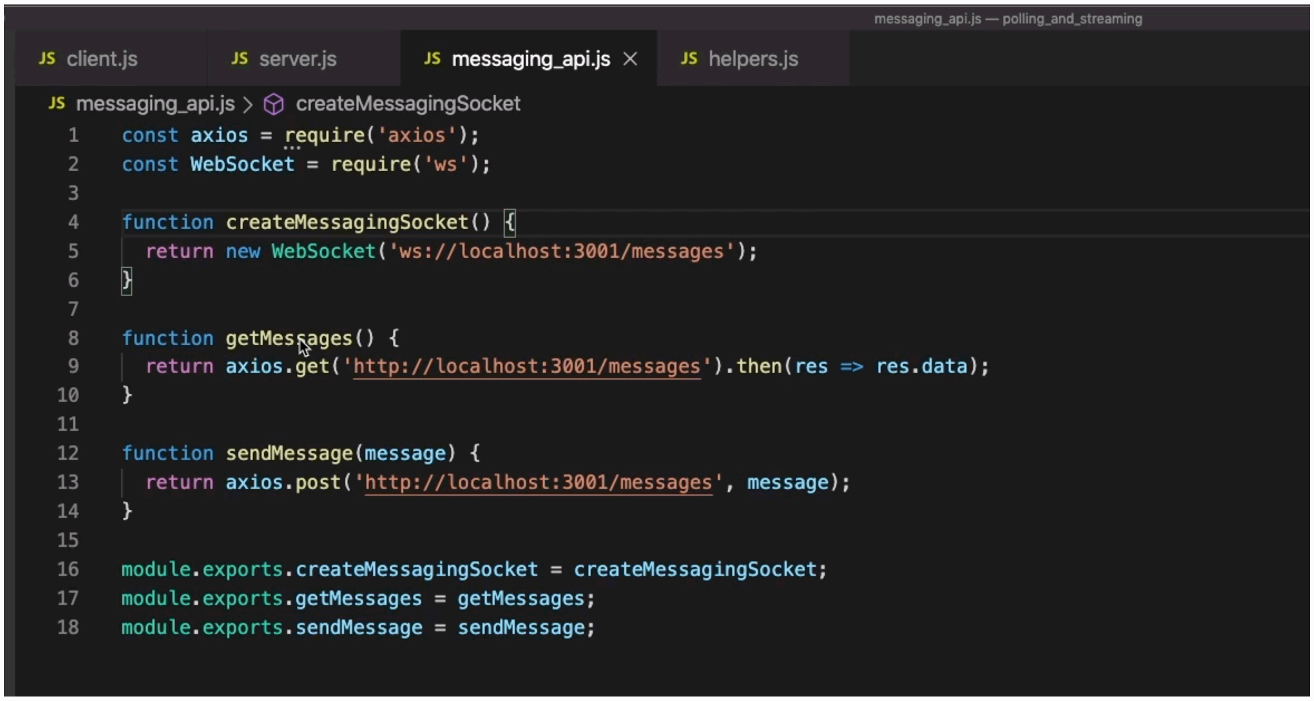
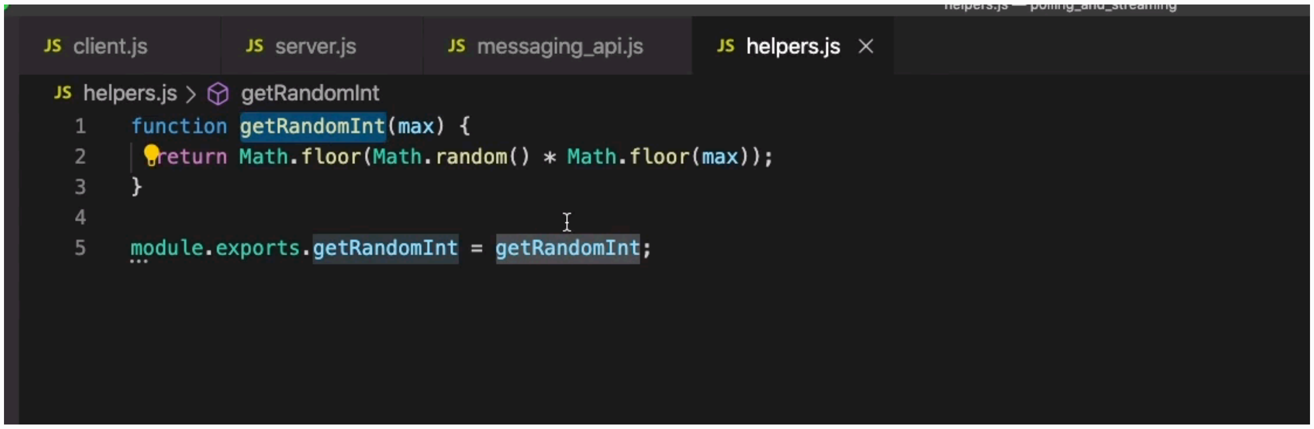
18_Polling and Streaming

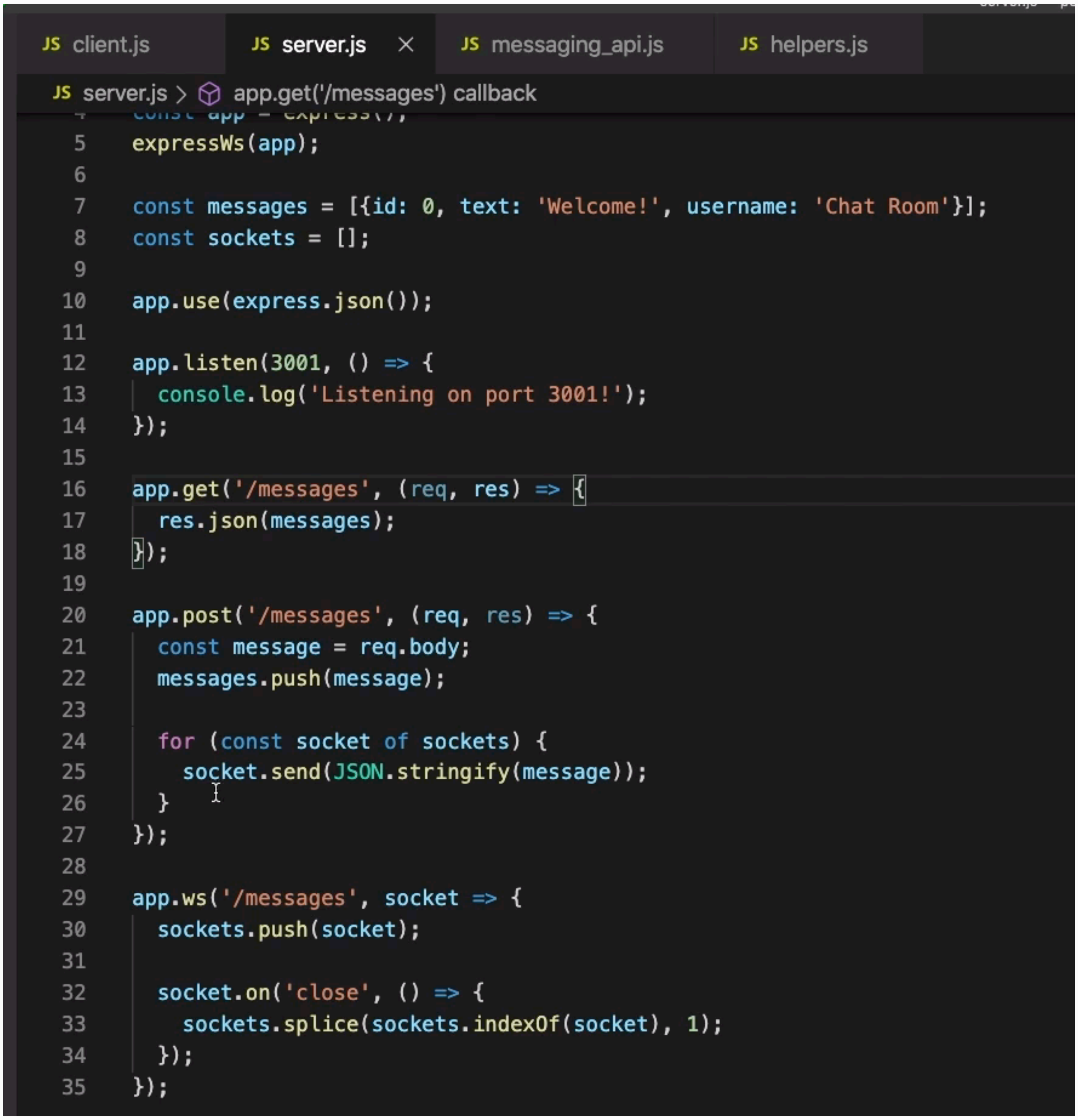



messaging_api.js — polling_and_streaming
client.js
server.js
messaging_api.js
helpers.js
messaging_api.js > createMessagingSocket

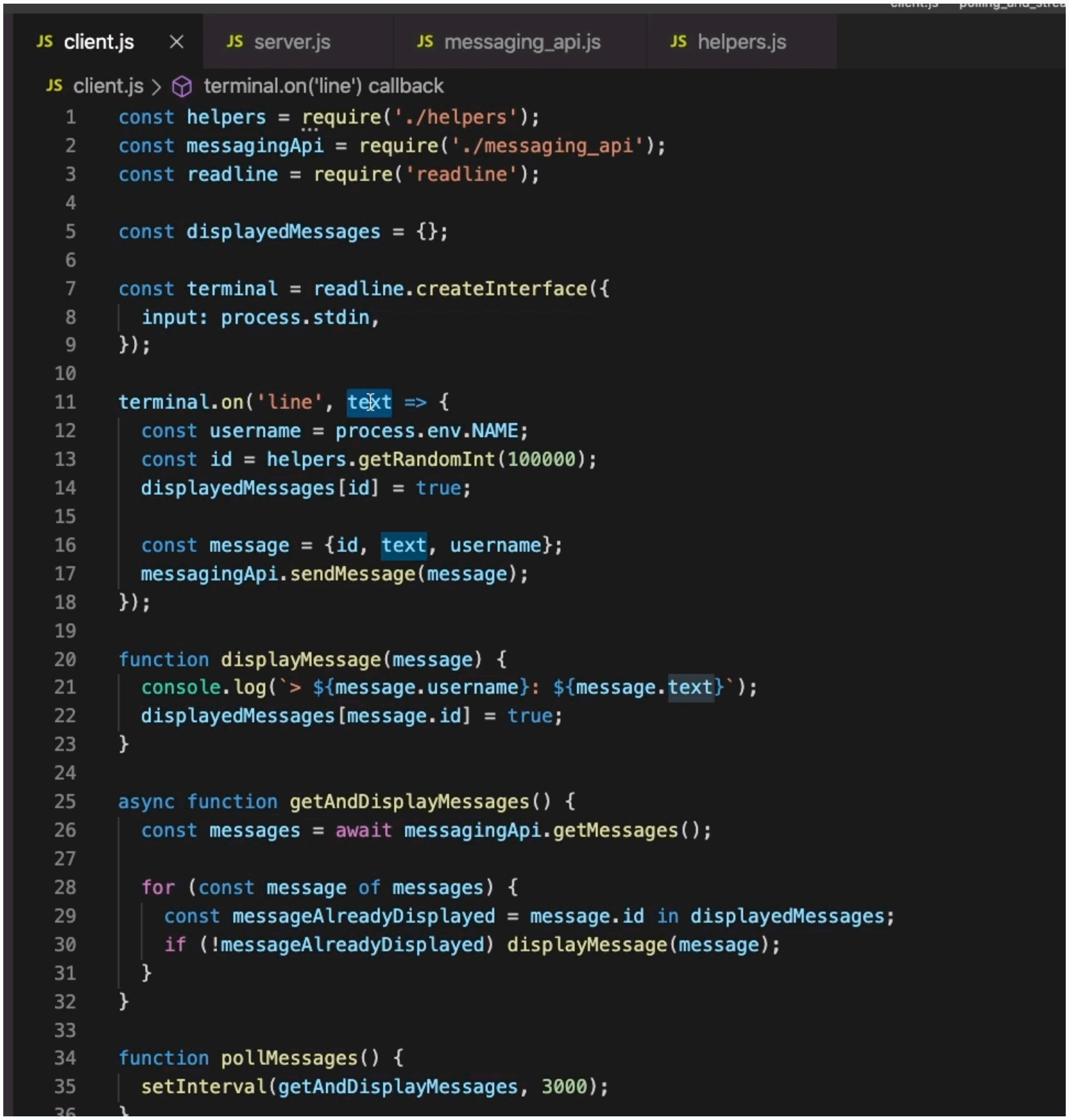
```javascript
const axios = require('axios');
const WebSocket = require('ws');

function createMessagingSocket() {
  return new WebSocket('ws://localhost:3001/messages');
}

function getMessages() {
  return axios.get('http://localhost:3001/messages').then(res => res.data);
}

function sendMessage(message) {
  return axios.post('http://localhost:3001/messages', message);
}

module.exports.createMessagingSocket = createMessagingSocket;
module.exports.getMessages = getMessages;
module.exports.sendMessage = sendMessage;
```



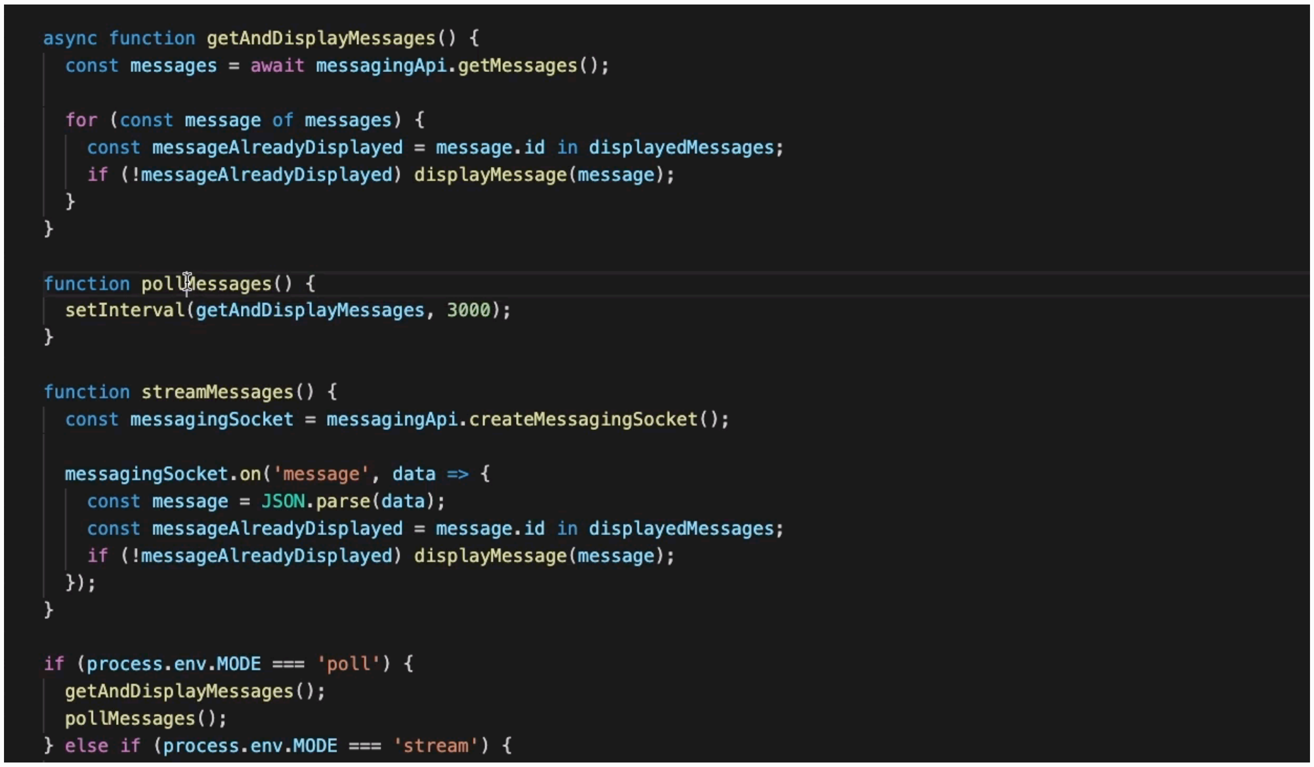
```javascript
function getRandomInt(max) {
  return Math.floor(Math.random() * Math.floor(max));
}

module.exports.getRandomInt = getRandomInt;
```



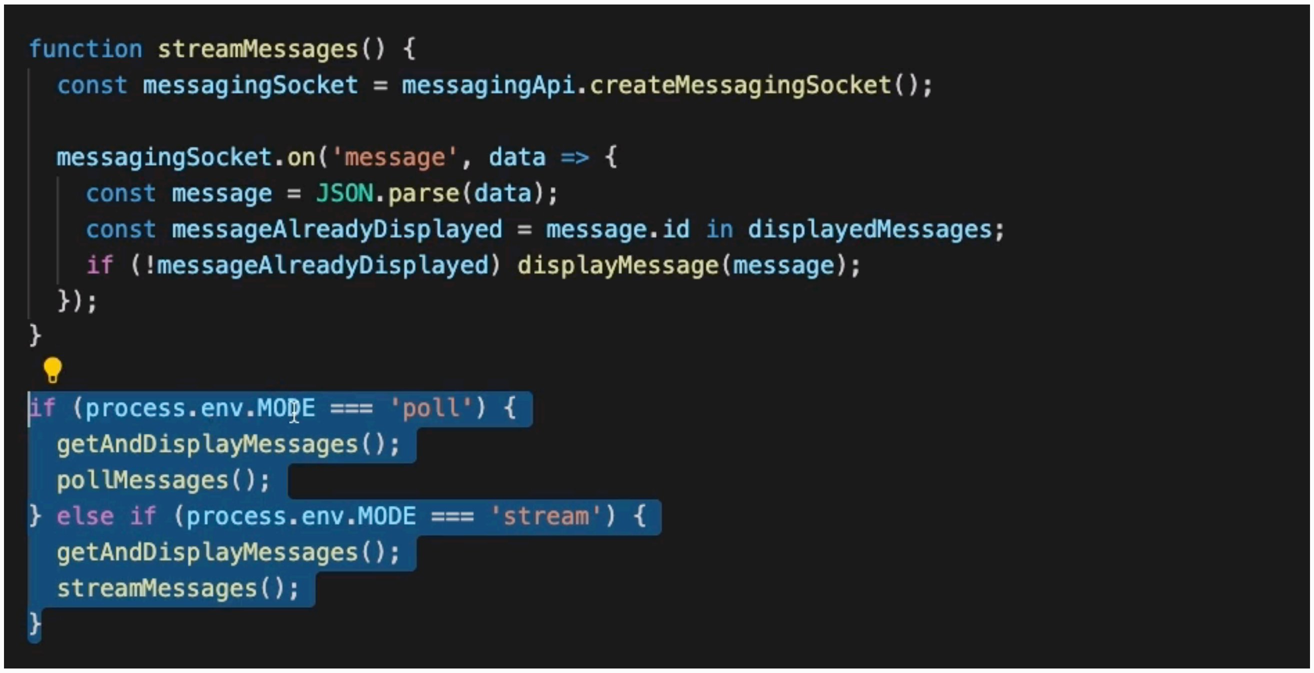
```javascript
expressWs(app);

const messages = [{id: 0, text: 'Welcome!', username: 'Chat Room'}];
const sockets = [];

app.use(express.json());

app.listen(3001, () => {
  console.log('Listening on port 3001!');
});

app.get('/messages', (req, res) => {
  res.json(messages);
});

app.post('/messages', (req, res) => {
  const message = req.body;
  messages.push(message);

  for (const socket of sockets) {
    socket.send(JSON.stringify(message));
  }
});

app.ws('/messages', socket => {
  sockets.push(socket);

  socket.on('close', () => {
    sockets.splice(sockets.indexOf(socket), 1);
  });
});
```



```javascript
const helpers = require('./helpers');
const messagingApi = require('./messaging_api');
const readline = require('readline');

const displayedMessages = {};

const terminal = readline.createInterface({
  input: process.stdin,
});

terminal.on('line', text => {
  const username = process.env.NAME;
  const id = helpers.getRandomInt(100000);
  displayedMessages[id] = true;

  const message = {id, text, username};
  messagingApi.sendMessage(message);
});

function displayMessage(message) {
  console.log(`> ${message.username}: ${message.text}`);
  displayedMessages[message.id] = true;
}

async function getAndDisplayMessages() {
  const messages = await messagingApi.getMessages();

  for (const message of messages) {
    const messageAlreadyDisplayed = message.id in displayedMessages;
    if (!messageAlreadyDisplayed) displayMessage(message);
  }
}

function pollMessages() {
  setInterval(getAndDisplayMessages, 3000);
```



```
async function getAndDisplayMessages() {
  const messages = await messagingApi.getMessages();

  for (const message of messages) {
    const messageAlreadyDisplayed = message.id in displayedMessages;
    if (!messageAlreadyDisplayed) displayMessage(message);
  }
}

function pollMessages() {
  setInterval(getAndDisplayMessages, 3000);
}

function streamMessages() {
  const messagingSocket = messagingApi.createMessagingSocket();

  messagingSocket.on('message', data => {
    const message = JSON.parse(data);
    const messageAlreadyDisplayed = message.id in displayedMessages;
    if (!messageAlreadyDisplayed) displayMessage(message);
  });
}

if (process.env.MODE === 'poll') {
  getAndDisplayMessages();
  pollMessages();
} else if (process.env.MODE === 'stream') {
```



```
function streamMessages() {
  const messagingSocket = messagingApi.createMessagingSocket();

  messagingSocket.on('message', data => {
    const message = JSON.parse(data);
    const messageAlreadyDisplayed = message.id in displayedMessages;
    if (!messageAlreadyDisplayed) displayMessage(message);
  });
}

if (process.env.MODE === 'poll') {
  getAndDisplayMessages();
  pollMessages();
} else if (process.env.MODE === 'stream') {
  getAndDisplayMessages();
  streamMessages();
}
```

2 Prerequisites

Client—Server Model

The paradigm by which modern systems are designed, which consists of clients requesting data or service from servers and servers providing data or service to clients.

Socket

A kind of file that acts like a stream. Processes can read and write to sockets and communicate in this manner. Most of the time the sockets are fronts for TCP connection.

2 Key Terms

Polling

The act of fetching a resource or piece of data regularly at an interval to make sure your data is not too stale.

Streaming

In networking, it usually refers to the act of continuously getting a feed of information from a server by keeping an open connection between the two machines or processes.