VIDEO CHAT APP- MESSAGING

What is our GOAL for this CLASS?

The goal of this module is to learn how socket.io can be implemented using Javascript.

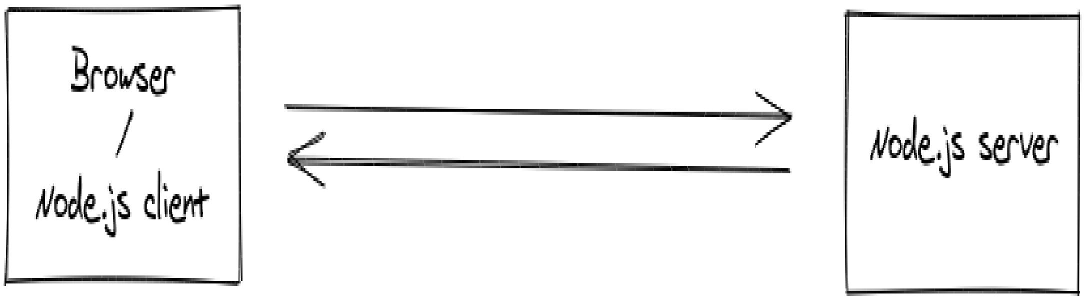
What did we ACHIEVE in the class TODAY?

- Learn about socket.io
- Implementation of socket.io using Javascript.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- socket.io

The KEY CONCEPT



SOCKET.IO

Socket.IO is a library that enables real-time, bidirectional and event-based communication between the browser and the server. It consists of:

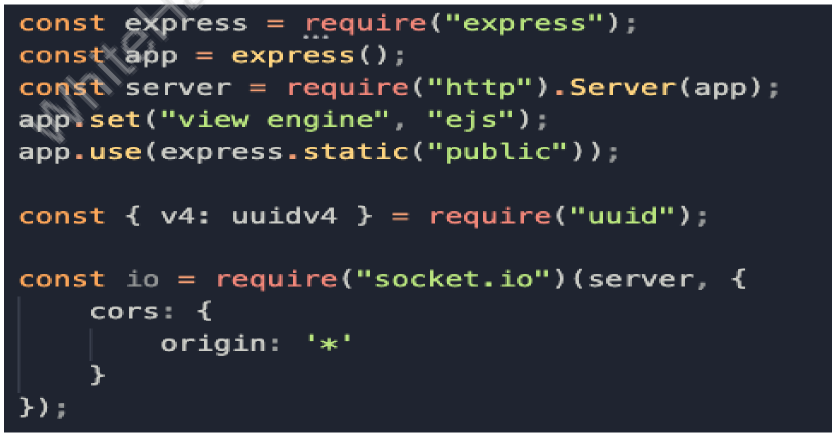
- a Node.js server
- a Javascript client library for the browser (which can be also run from Node.js).

How did we DO the activities?

In the last class, we learnt about NodeJs , integrated HTML code into NodeJS server and made all pages have a unique id. Today, we learnt about socket.io.

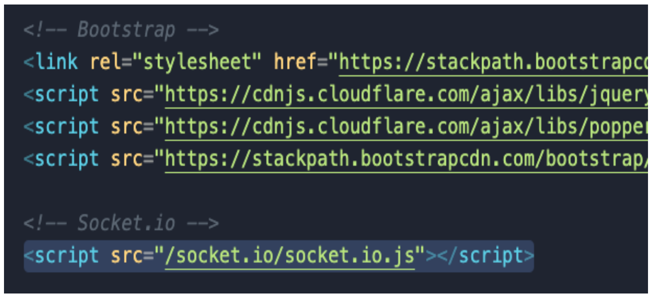
Activity:

1. In the code that we completed in C-215 ,Import socket.io in ***server.js*** file. Create a constant "io" and use require to include socket.io. After requiring it, define the server in which you want to use it.
   In the options, we are setting the ***cors***' ***origin*** to **"*"**, with this we tell the browser to allow sharing data from all the ports instead of blocking it.



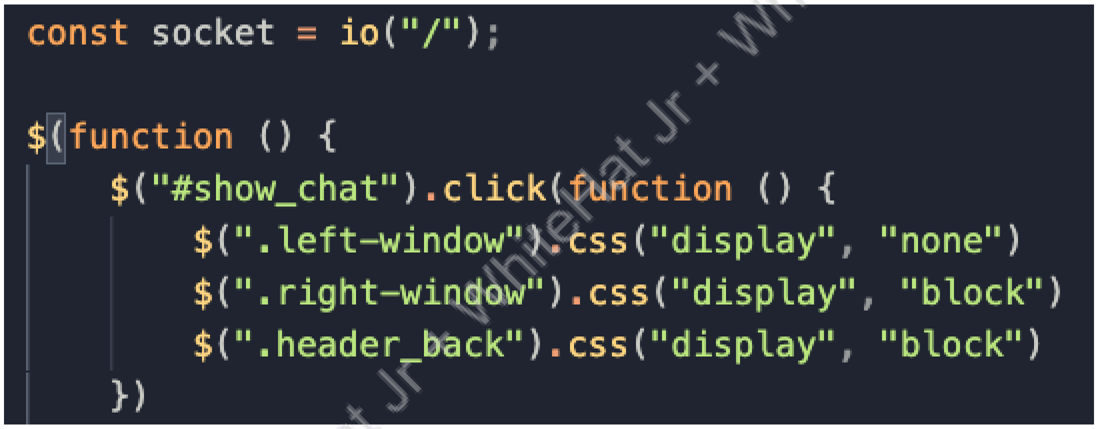
```
const express = require("express");
const app = express();
const server = require("http").Server(app);
app.set("view engine", "ejs");
app.use(express.static("public"));

const { v4: uuidv4 } = require("uuid");

const io = require("socket.io")(server, {
    cors: {
        origin: '*'
    }
});
```

2. Let's add a script to import socket.io in our client, in our ***index.ejs*** to import socket.io



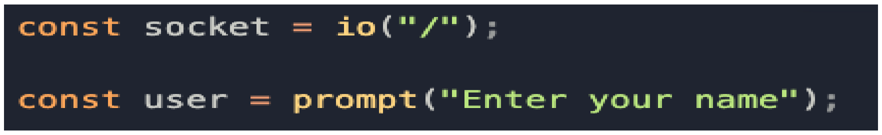
```html
<!-- Bootstrap -->
<link rel="stylesheet" href="https://stackpath.bootstrapc
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper
<script src="https://stackpath.bootstrapcdn.com/bootstrap/

<!-- Socket.io -->
<script src="/socket.io/socket.io.js"></script>
```

3. In script.js, use socket.io to create a socket. To create a socket at client side in HTML code, use socket.io in the io("/") function.



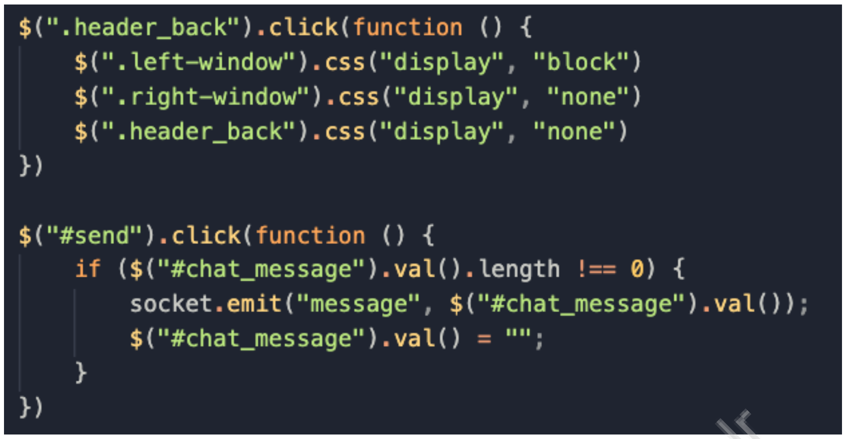
```javascript
const socket = io("/");

$(function () {
    $("#show_chat").click(function () {
        $(".left-window").css("display", "none")
        $(".right-window").css("display", "block")
        $(".header_back").css("display", "block")
    })
```

4. Let's use the prompt function(which make the browser ask the user a question and save their response in a variable.) We will make the changes in ***script.js*** -



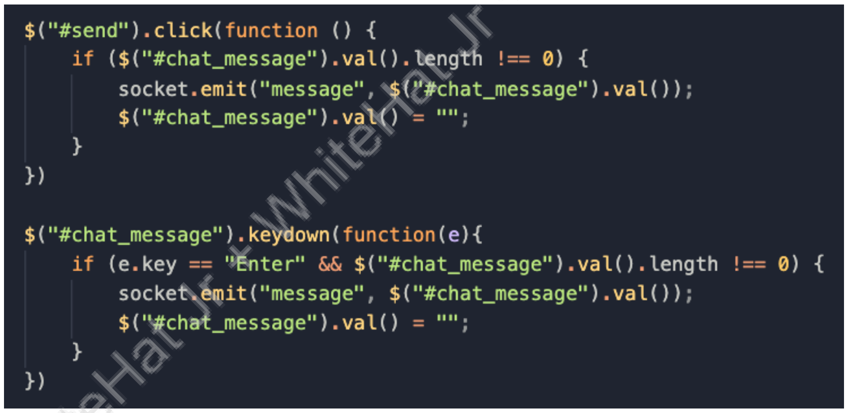
```javascript
const socket = io("/");

const user = prompt("Enter your name");
```

5. Let's use id "send" to create an event listener in script.js.



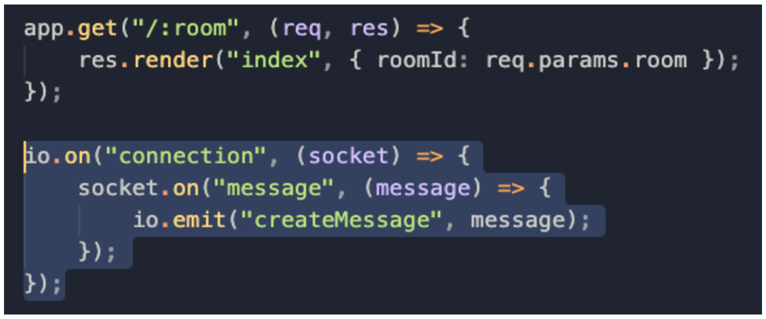
```
$(".header_back").click(function () {
    $(".left-window").css("display", "block")
    $(".right-window").css("display", "none")
    $(".header_back").css("display", "none")
})

$("#send").click(function () {
    if ($("#chat_message").val().length !== 0) {
        socket.emit("message", $("#chat_message").val());
        $("#chat_message").val() = "";
    }
})
```

6. To allow users to use the “Enter” key to send messages. Let’s create an event using keyDown().



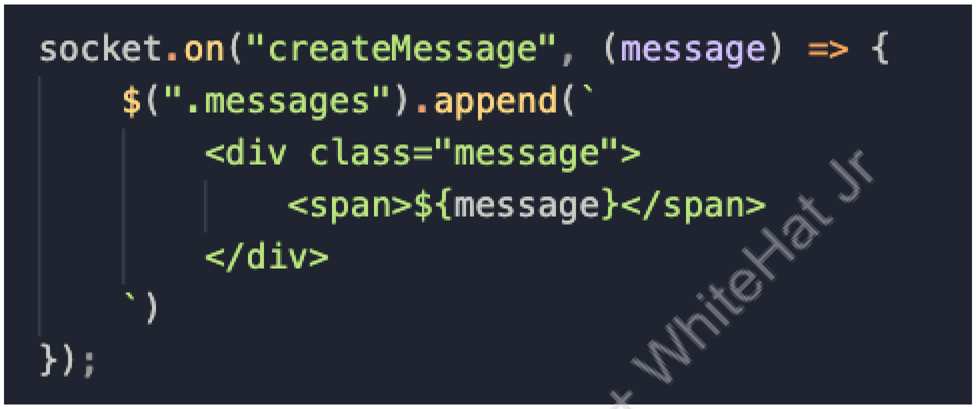
```
$("#send").click(function () {
    if ($("#chat_message").val().length !== 0) {
        socket.emit("message", $("#chat_message").val());
        $("#chat_message").val() = "";
    }
})

$("#chat_message").keydown(function(e){
    if (e.key == "Enter" && $("#chat_message").val().length !== 0) {
        socket.emit("message", $("#chat_message").val());
        $("#chat_message").val() = "";
    }
})
```

7. Now, we have already emitted the message from our client, and we want to receive and emit it back to all the clients from the server. Let’s add code in server.js.



```
app.get("/:room", (req, res) => {
    res.render("index", { roomId: req.params.room });
});

io.on("connection", (socket) => {
    socket.on("message", (message) => {
        io.emit("createMessage", message);
    });
});
```

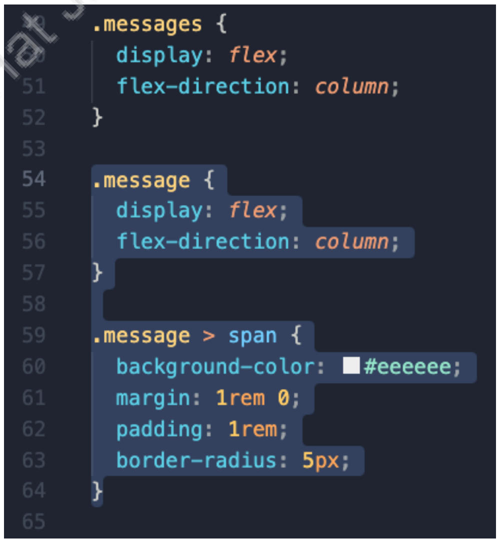
8. The ***createMessage*** socket event we will create at the client side, to render the message in the chat box.
   Let's write the code for it in our ***script.js*** now -



```
socket.on("createMessage", (message) => {
    $(".messages").append(`
        <div class="message">
            <span>${message}</span>
        </div>
    `)
});
```

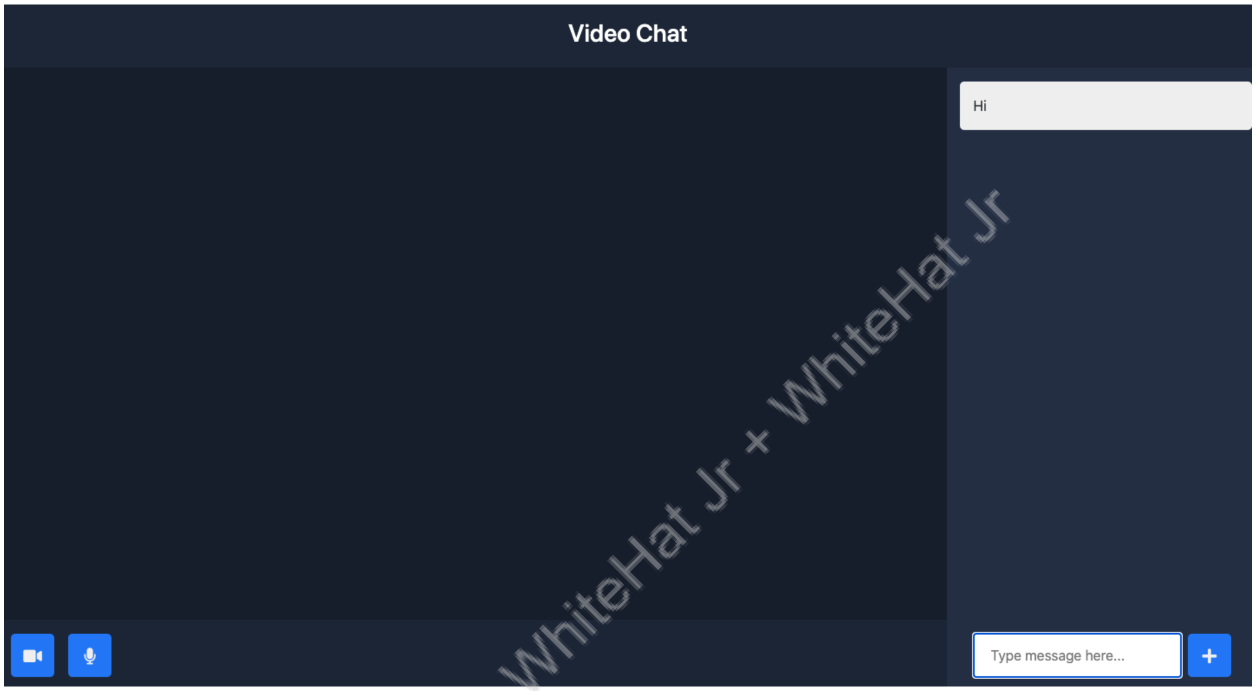
9. To test run the server with ***npm start*** command in the command prompt / terminal and open ***localhost:3030*** in the browser.

10. Let's add some styling for it in ***style.css***.



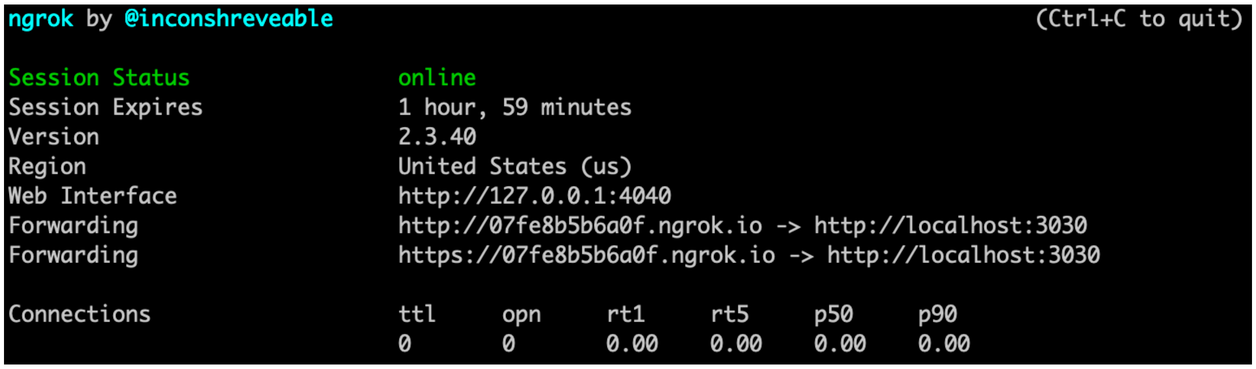
```
.messages {
  display: flex;
  flex-direction: column;
}

.message {
  display: flex;
  flex-direction: column;
}

.message > span {
  background-color: #eeeeee;
  margin: 1rem 0;
  padding: 1rem;
  border-radius: 5px;
}
```

11. Now, refresh the page.



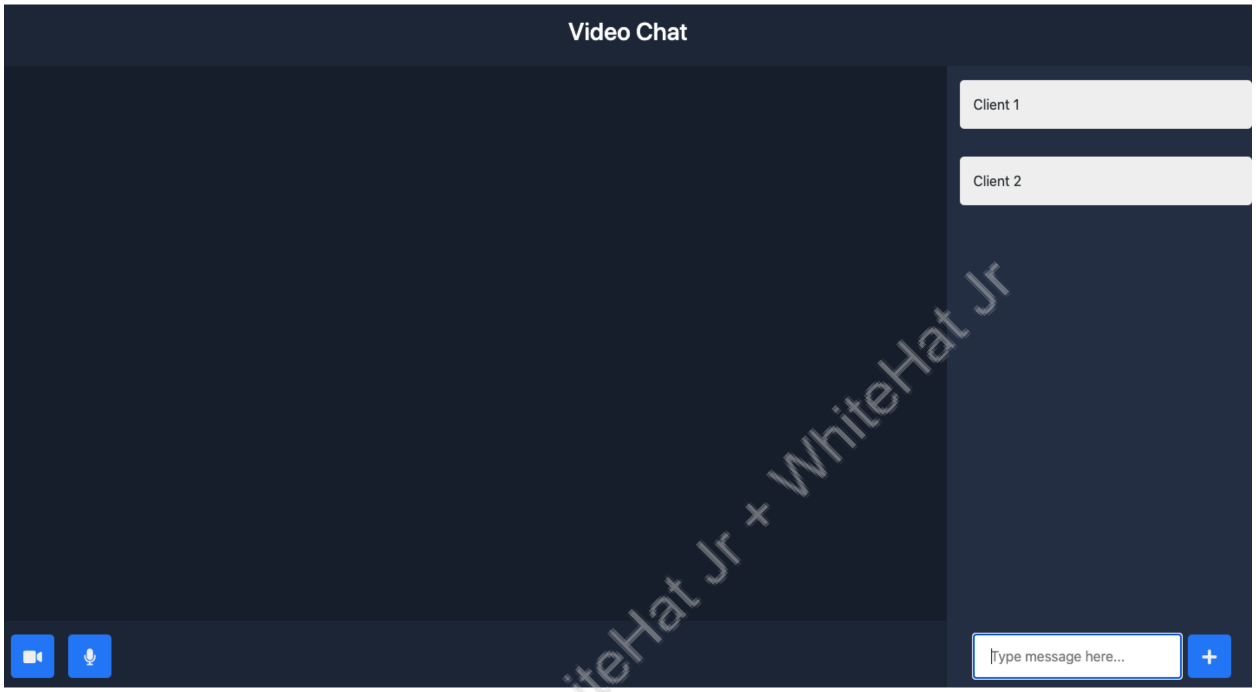
12. Let's try to run the ngrok server in a new command prompt / terminal.

```
ngrok http 3030
```



```
ngrok by @inconshreveable                                       (Ctrl+C to quit)

Session Status                online
Session Expires               1 hour, 59 minutes
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://07fe8b5b6a0f.ngrok.io -> http://localhost:3030
Forwarding                    https://07fe8b5b6a0f.ngrok.io -> http://localhost:3030

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

13. Copy the HTTPS URL, and paste it in the browser.



What's NEXT?
In the next class, we will learn about PeerJS.

Expand Your Knowledge:
Explore more about messaging here.