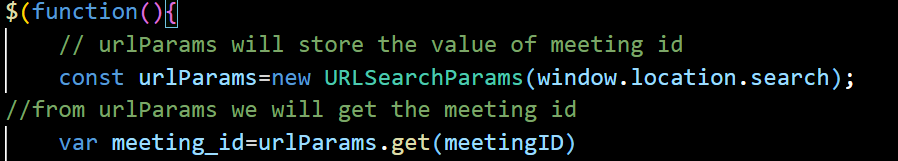
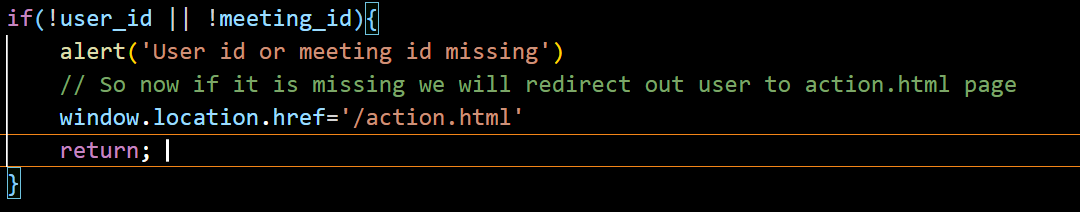
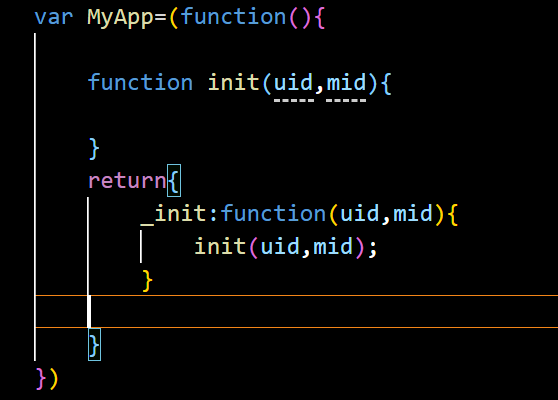
Section 4

1. Basic Js setup

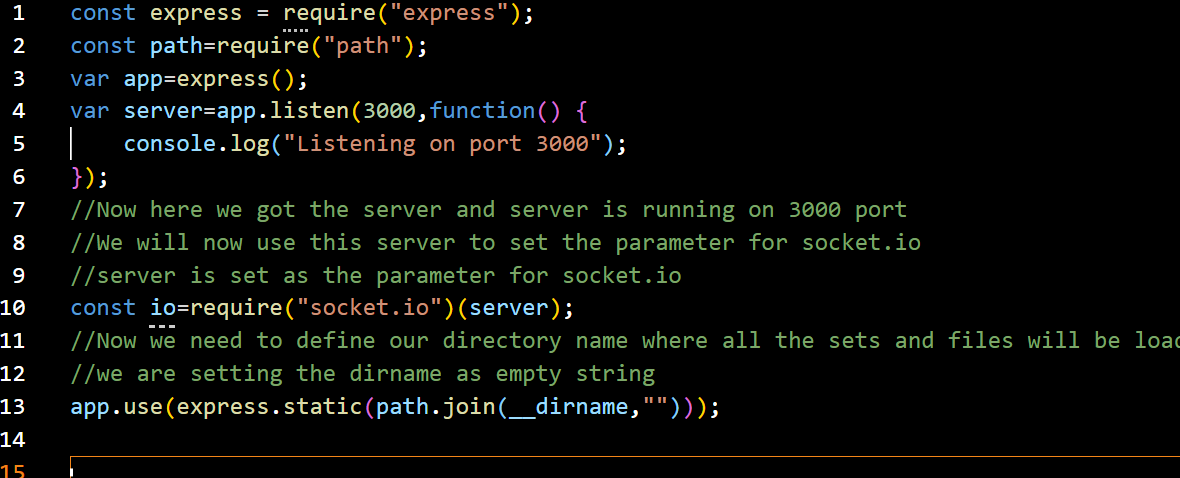


* First urlParams will store the data from the generated link and from this we will get out meeting id
* Also we will try to fetch the username which is entered by the user
* 
* Now if the username or meeting id is missing then we will not allow to enter the web conferencing page
* 

* Now else if it is entered then we have display meetingContainert
* 
* Now we will create another js file in which we are going to store our userid and meeting id

})

1. Server Setup

* Now we will be working on server.js file
* 

The code snippet you provided is using the Express.js framework along with the Socket.IO library to create a server that serves static files.

Here's a breakdown of the code:

1. The first line imports the `express` module.

2. The second line imports the `path` module.

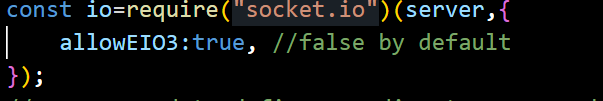
3. An instance of the Express application is created and stored in the `app` variable.

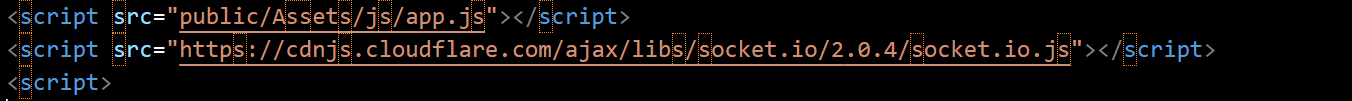
4. The `listen` method is called on the `app` object to start the server on port 3000. A callback function is provided to log a message indicating that the server is listening.

5. The `socket.io` library is imported, and the `server` object (created by the `app.listen` method) is passed as a parameter to the `socket.io` function to initialize the Socket.IO server.

6. The `express.static` middleware is used to serve static files. The `path.join` function is used to construct the absolute path to the directory containing the static files. In this case, it uses `\_\_dirname` (which represents the current directory) and an empty string to indicate that the static files are located in the same directory as the script file.

With this code, any files in the current directory can be accessed via the server. For example, if there is a file named `index.html` in the same directory, it can be accessed at `http://localhost:3000/index.html`.



* This should be added for the new version so that socket id can be printed
* First close the live server then in terminal write npm start
* 3005:port number
* Now in browser write <http://localhost:3005/action.html>
* Part 3
* Now we will add socket id to client
* Now we will create a app.js file for this
* 
* Now after doing this we need to apply it to the index.html also
* 

The code snippet you provided is an example of a JavaScript module pattern, specifically the Immediately Invoked Function Expression (IIFE) pattern. It encapsulates the code within an anonymous function that is executed immediately.

Here's how the code works:

1. The `MyApp` variable is declared and assigned the result of an IIFE.

2. Inside the IIFE, the `init` function is defined. It takes two parameters `uid` and `mid`.

3. The `init` function calls the `event\_process\_for\_signaling\_server` function.

4. The `socket` variable is declared and initialized with `null`. It will hold a reference to the socket object.

5. The `event\_process\_for\_signaling\_server` function is defined. It establishes a connection to a signaling server using the `io.connect()` method. It then listens for a "connect" event on the socket and displays an alert when the connection is successfully established.

6. Finally, an object is returned from the IIFE, exposing a single method `\_init` as a public interface. This method can be used to initialize the application by calling the `init` function with the provided `uid` and `mid` parameters.

The purpose of this code appears to be setting up a client-side application that connects to a signaling server using Socket.IO. The `init` function is called with a `uid` and `mid` parameter to initialize the application, and the connection to the signaling server is established in the `event\_process\_for\_signaling\_server` function.