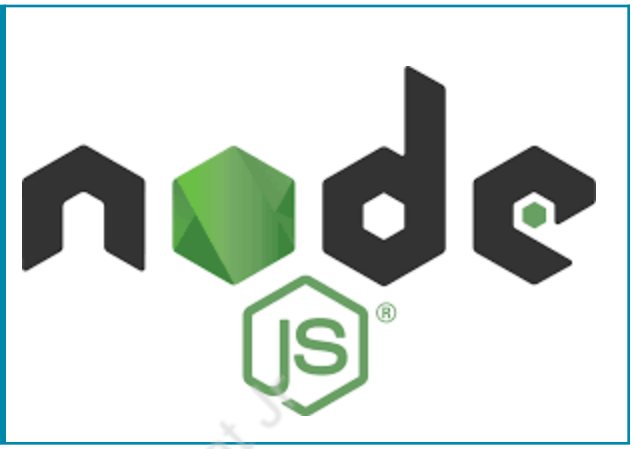
VIDEO CHAT APP - NodeJS



What is our GOAL for this CLASS?

The goal of this module is to learn about NodeJS, ExpressJS and creating a server to host HTML files.

What did we ACHIEVE in the class TODAY?

- Integrate HTML code into NodeJS server.
- A unique ID should be assigned to all pages.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- NodeJS

The KEY CONCEPT

- **NodeJS**

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project.

It can be used to create a backend server.
Node.js has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language.

How did we DO the activities?

Activity:

1. Create an empty NodeJS project. Create a new directory called ***video-chat-app*** and navigate into this directory.

```
mkdir video-chat-app
cd video-chat-app
```

2. To create an empty project template for NodeJS, we can run the following command -

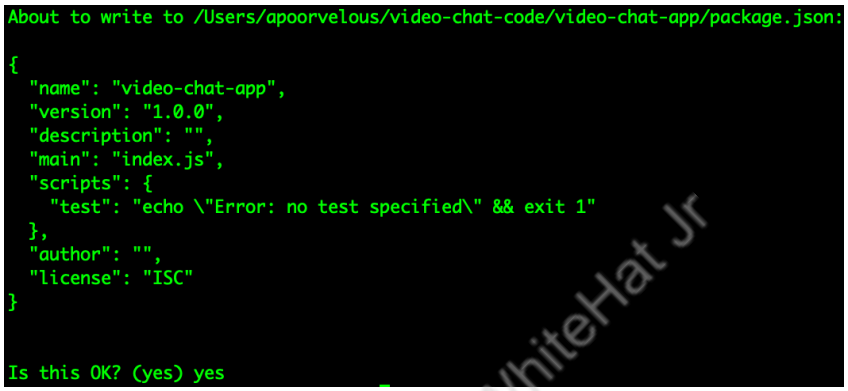
```
npm init
```

- On running the above command in the project directory, there are a couple of things that you will be asked to confirm for the project -

```
Press ^C at any time to quit.
package name: (video-chat-app)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
```

- Press “Enter” as we don’t want to change the configuration of the project. It will then tell you how your ***package.json*** file would look like. Enter ***yes*** in the prompt and press enter.

- It should now create the project’s ***package.json*** file for you.

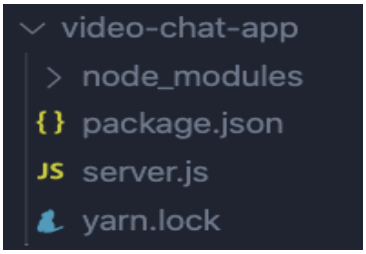


3. Use the following command to install a few libraries into this project

```
yarn add express ejs socket.io uuid peer
```

- This command will install five most important libraries that we need to build our project -
    - ExpressJS
    - EJS
    - Socket.io
    - UUID
    - Peer

4. Open our project in VS Code editor and create a new file called ***server.js.***



5. Import our ***express.js***, which is our framework that we will use to create our

server in NodeJS.

```
const express = require("express");
const app = express();
const server = require("http").Server(app);
```

6. Next, we are creating our "server app" with the ***express()*** function that we just require it in our project, and saving it into another constant called ***app.***

```
const express = require("express");
const app = express();
const server = require("http").Server(app);

app.get("/", (req, res) => {
    res.status(200).send("Hello World");
});
server.listen(3030);
```

- A **get** request on the app with the ***app.get()*** notation, inside which, we are first writing a route on which it will work
- Two arguments are mandatory for all the APIs of any type that you create in NodeJS.
- ***req*** stands for request, and contains all the data that is sent in the request (mostly used in POST requests where you are saving some data into the API), and a ***res*** keyword, stands for Response, which is the response that this API will send back. The server will listen on port 3030, with ***server.listen(3030)***

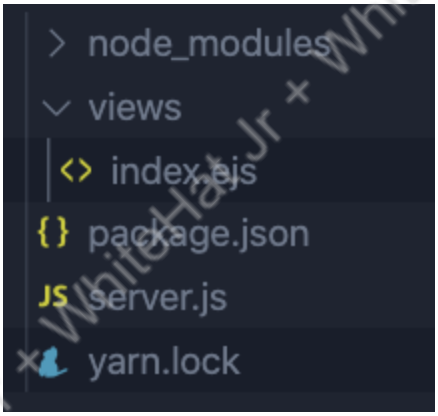
7. Run the following command.

```
npm start
```

8. To display it through ***EJS*** so we can add some logic to our HTML, we will have to set our ***view engine*** of the app to ***ejs***, so that NodeJS knows that we are using EJS for rendering our HTML!

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

9. Create a view ***folder*** and inside that folder, we can create a file called ***"index.ejs".***



10. Now, The first thing that we would want to do is to copy our ***index.html*** code from the last class into the index.ejs *file*.

index.ejs

213t > views > index.ejs > html

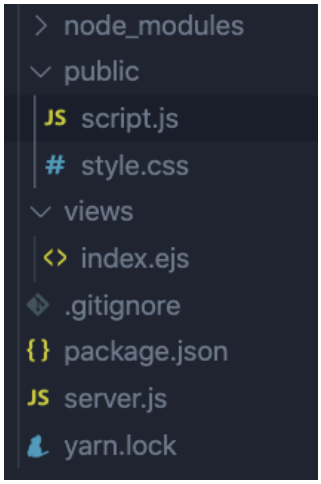
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Video Chat App</title>
    <link rel="stylesheet" href="style.css" />
    <script src="https://kit.fontawesome.com/c939d0e917.js"></script>

    <!-- Bootstrap -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></scr
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script
</head>

<body>
    <div class="row" style="overflow: hidden;">
        <div class="col-sm-12 col-md-12 col-lg-12 text-center p-3" style="background-color: #1d
            <div class="header_back">
                <i class="fas fa-angle-left"></i>
            </div>
            <h3 class='text-white'>Video Chat</h3>
        </div>
    </div>
    <div class="row main">
        <div class="col-sm-12 col-md-12 col-lg-9 left-window">
            <div class="row">
                <div class="col-sm-12 col-md-12 col-lg-12" style="height: 81vh; background-color:
```


11. Create a folder called ***public*** and have our ***style.css*** and ***script.js*** in this folder.



12. Now we need to define in our NodeJS **server.js** file, that our public folder is hosting static data.

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

- We are letting our app know that it needs to ***use*** the ***"public"*** folder to host the ***static*** data with the ***express.static()*** function

13. In order to differentiate different rooms in the video app we need unique URLs. We will now require ***uuid*** that we installed earlier into our ***server.js***, with the following line of code.

```
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");
```

14. We want to make sure that when someone comes to the URL ***"/"***, we are redirecting them to a unique URL, so our code in the ***"/"*** API would also change.

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

app.get("/", (req, res) => {
    res.redirect(`/${uuidv4()}`);
});
```

15. We also want to display our **"*index.ejs*"** file on this URL, so that the user sees the client side frontend code.
    Let's create a new API for that!

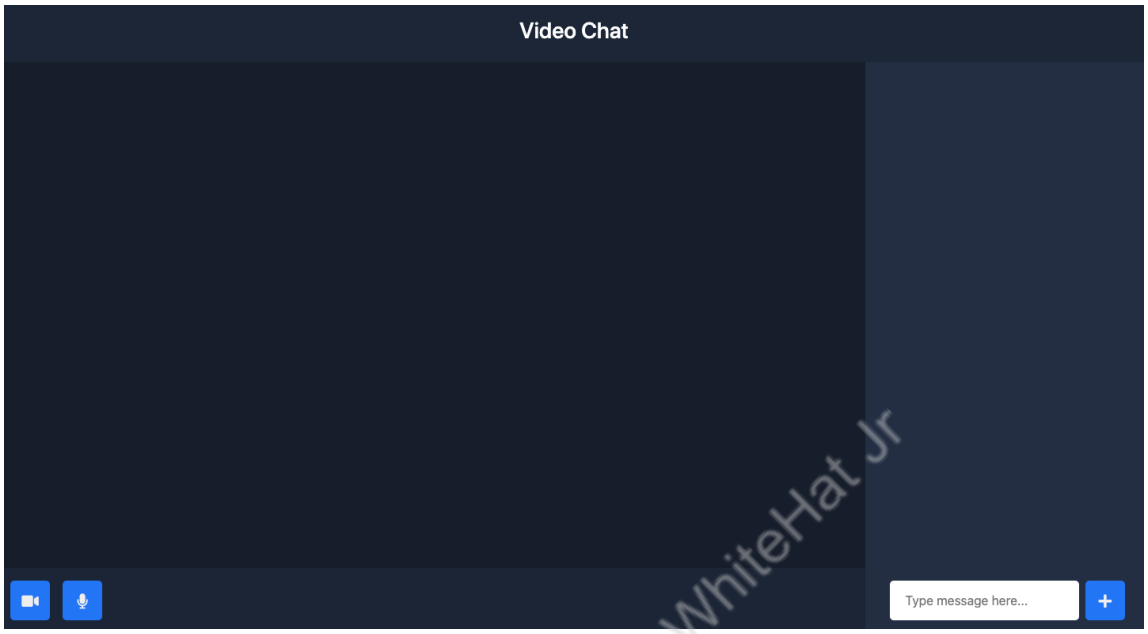
```
app.get("/", (req, res) => {
    res.redirect(`/${uuidv4()}`);
});

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


We created a new ***GET API*** with our app on URL ***"/:room",*** where we know that the ***room*** here is the unique ID of the room.

API, we are using the ***res.render()*** function this time, to render our ***index.ejs*** file. While rendering it, do note that we are also sending the ***room***, which is our unique room ID in a variable called ***roomId*** to the client.

16. Let's check the output by starting the server again using command ***npm start*** command.



What's NEXT?

In the next class, we will learn about messaging.

Expand Your Knowledge:

You can learn more about NodeJS from https://nodejs.dev/learn .