Workshop on Node.js

The objective of this workshop to allow you to understand how node render html files, then make use of APIs to manage data in designed JSON files.

We will be using Visual Studio Code (VS code) as a tool to build up the entire product.

**Part 1: Start a project**

Create a new folder MSBA23\_node with VS code

A screenshot of a computer

Description automatically generated

Create a new terminal,

A screenshot of a computer

Description automatically generated with medium confidence

Issue the command ***npm init***, press enter to all options.

Text

Description automatically generated

Ensure that after this, the package.json is automatically created. Take a look at the package.json file.

Graphical user interface, application

Description automatically generated

**Part 2: install the dependencies**

Issue the command ***npm i express***, press enter to all options.

Text

Description automatically generated

A picture containing text, scoreboard, screenshot

Description automatically generated

Issue the command ***npm i -- save-dev dotenv nodemon***Text

Description automatically generated

Create a .gitignore file as follows :

Graphical user interface, application

Description automatically generated

Amend package.json file :

Text

Description automatically generated => Graphical user interface, text

Description automatically generated

**Part 3: Get ready the server**

Copy the following codes to a new file call server.js:

const express = require('express')

const app = express()

const port = 3000

app.listen(port, () => {

  console.log(`Example app listening on port ${port}`)

})

Activate the server with npm run devStart. (Recall devStart is the code we amended in package.json)Text

Description automatically generated

**Part 4: Preparing the routes**

Create a folder call router and create a new .js file call player.js

Graphical user interface, text, application, chat or text message

Description automatically generated

Prepare the player.js initial routing requirements as follows:

const express = require('express')

const router = express.Router()

module.exports = router

Include this route into the sever.js

const express = require('express')

const app = express()

const port = 3000

const playRouter = require('./routes/player')

app.use('/player', playRouter)

app.listen(port, () => {

  console.log(`Example app listening on port ${port}`)

})

**Part 4: Preparing the REST APIs**

In the players.js, include the codes below:

const express = require('express')

const router = express.Router()

let playerNow =1

// Getting all

router.get('/', (req,res)=>{

    res.send(`Player ${playerNow}, your turn`)

})

module.exports = router

Now, open the browser, then enter <http://localhost:3000/player>

Amend the code above to the following:

const express = require('express')

const router = express.Router()

let playerNow =1

router.get('/now', (req,res)=>{

    res.send(`Player ${playerNow},your turn`)

})

router.get('/next', (req,res)=>{

    playerNow = 3-playerNow

    res.send(`Next please, Player ${playerNow}`)

})

module.exports = router

Try the following command in the browser and observe the output

<http://localhost:3000/player>

<http://localhost:3000/player/now>

<http://localhost:3000/player/next>

<http://localhost:3000/player/now>

**Part 5: Render HTML Page in Express**

Create a index.html in the folder:

Enter “ht”, you will the intelli-sense appear,

A screenshot of a computer

Description automatically generated

select html:5 and type enter, observe what you see

A screenshot of a computer

Description automatically generated

Create 2 buttons btnAbout and btnAdd.

Modify the codes above to be as follow:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <button id="addNew">Add New</button>

<button id="aboutMe">About Me</button>

</body>

</html>

Inside **server.js** file add the following code:

const path = require('path')

app.get('/main', (req,res)=>{

    res.sendFile('index.html',{

        root: path.join(\_\_dirname,"./")

    })

})

Try the following command in the browser and observe the output

<http://localhost:3000/main>

Graphical user interface, text, application

Description automatically generated

Create a folder call client

Create a page call aboutMe.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p> about Me ....<p>

</body>

</html>

Then in the server include :

// For serving static HTML files

app.use(express.static("client"));

To access any html in the client folder, you may use similar code as below :

http://localhost:3000/aboutme.html

**Part 6: Complete the game**

Amend game.html to be as follows :

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>2 players game</title>

</head>

<body>

    <p id="p1">starting</p>

    <button onclick="next()">See if this work</button>

</body>

<script>

    window.addEventListener('load', init);

    function init() {

    const p1 = document.getElementById('p1');

    const requestURL = 'player/now';

    fetch(requestURL)

    .then(res => res.text())

    .then(data =>     {

        const p2 = document.getElementById('p1');

        p2.innerHTML = data

    })

    .catch(err => console.error(err))

   }

   function next() {

    const requestURL = 'player/next';

    fetch(requestURL)

    .then(res => res.text())

    .then(data =>     {

        const p2 = document.getElementById('p1');

        p2.innerHTML = data

    })

    .catch(err => console.error(err))

   }

  </script>

</html>